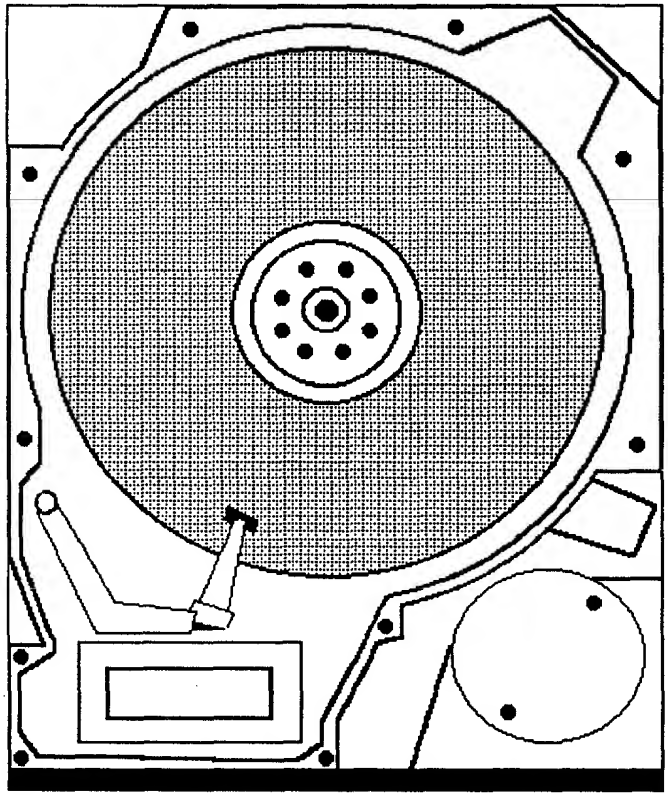
the world of 68' micros

Support for Motorola Processors

01 May 1994 | Vol. 1 Number 7 | $4.50 Canada, $4.00 US

what you've been waiting for...

the HARD DRIVE issue!



History, how, and why of hard drives

* *Hard Drives for OSK*
* *Ken-Ton SCSI Interface- Hard Drive for CoCo*
* *Good Sources for Hard Drives - Cheap!*
* *Next Issue - Burke&Burke CoCo HD Interface!!*

CONTENTS

POSTMASTER:
If undeliverable return to:
FARNA *Systems* PB
Box 321
Warner Robins, GA 31099

Address Correction Requested

the world of 68' micros

Published by:
FARNA Systems
P.O. Box 321
Warner Robins, GA 31099-0321

Editor: F. G. Swygert

Subscriptions:
$23/year (8 issues) US; $30/year for Canada/Mexico ($12 US, $16 C/M for six months- four issues). Overseas $35/year ($18 for four issues) surface. Add $8/year, $4 six months for air mail.
microdisk: $40 per year, $21 six months, or $6 per issue. Overseas add $10/year, $5/six months, $1/single issue for air mail delivery. microdisk contains programs and source listings from each magazine; not stand-alone.

Advertising Rates:
$15 1/6 page, $20 1/4 page, $35 1/2 page, $60 full page, copy ready. Add $10 for special placement, $10 for typesetting ($5 1/4 or less). Dot matrix will be typeset if deemed unacceptable and submitter billed. 10% discount for four or more appearances.

The publisher welcomes any and all contributions. Submission constitutes warranty on part of the author that the work is original and not copywritten by another party. All opinions expressed herein are those of the individual writers, not necessarily the publisher or editor. FARNA Systems reserves the right to edit or reject any submitted material without explanation. Renumeration discussed on an individual basis.

Back issues are $4 per copy. Overseas add $1 each for airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via **Internet** (dsrtfox@Delphi.com) or **Delphi** E-mail (DSRTFOX). The CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are sponsored by **Falsoft**).

The editor speaks...

F.G. Swygert

This issue...

I've been promising a hard drive issue for some time- and here it is! Everyone should get something from the feature article. OSK computers and the Ken-Ton CoCo SCSI controller are covered in this issue. The Disto and CoCoXT from Burke&Burke will be discussed in the next couple issues. I also found some info on the new IDE type drives that should prove interesting... that will appear in a later issue also. Have to save some room for the regular columns and such!

Don't forget to renew subscriptions!

I'll be sending postcards out for renewal notices soon. You won't get one if you've already renewed, since I'll be going by the date on the mailing labels. The numbers after your name is the date of the last issue of your current subscription. "0794" or 07/94" means the last issue will be in August of 1994. Yes, I know... it should be "08/94", but for some of the first subscriptions sent in I used 07 instead of 08. That will be corrected as renewals come in. So do remember to check your expiration date and consider renewing.

Let us know what you want...

I believe it's time I get a report on how the magazine is doing before we start another year together. I'll have a questionnaire insert in this issue. Please take the time to fill it out and send it back in. I need your input to make the magazine better serve you, the reader. After all, it is YOUR magazine as much as my own. So do your part and let me know what you think needs to be done.

PROGRAMMING CONTEST!!!

I'll be printing program contest winners from our "sister" publication in Australia (CoCo-Link). This got me thinking that maybe I should sponsor a contest also. **I'm going to earmark some cash for the overall winner... let's say $50.** This will be for the overall most impressive piece of work submitted. **There will be a $25 cash prize for the overall second place winner.** There will be first and second place prizes in three categories: OSK/OS-9000; OS-9; and DECB. First place prizes will be a copy of "Tandy's Little Wonder" or up to $18 in other FARNA Systems products, second place will be a QRG (OSK or OS-9) or up to $10 in FARNA Systems products. ***That's over $150 in prizes!***

The rules are simple: create your best program in whatever language you prefer. Send it in with a description of what equipment is required to run it and what it does. Documentation and source code is necessary. The author retains copyright, but grants FARNA Systems license to print/reprint as it sees fit. Winners will be printed first, all submissions are subject to be printed. **Entries should be submitted no later than 01 September 94.** That gives you four months to work. Winners will be announced in the 15 December 94 or 01 February 95 issue, depending on when judging is completed.

Programs will be organized into categories as they are received. They will then be judged by at least two people, three whenever possible. Judging will be on a point system, 1 being worst, 10 being the best. At least three people will judge each item. Each judge will write a short review of the program(s) judged. The winner in that category will be chosen by averaging the three scores. The written reviews will be used to determine overall winners. Hardware requirements will determine judges. I will have the final say as to overall winners.

Prizes for various categories were considered, but decided against. Games will be judged right along with utilities and applications, OS-9 and OSK along with DECB. The main reason for this is that I couldn't cover all categories with decent prizes. Just try to impress the judges with your talent! A DECB game could easily win over an OSK or OS-9 game, utility, or application (DECB is easier to program games in anyway), but then a simple utility that makes a difficult task easy could be more impressive than any game. Use your imagination and show off your skills! And remember, simplicity (and neat code) has it's own beauty...

You'll find that there is no longer a section with the contents of "microdisk" published. It was just to much to update that section while trying to get the magazine out on time! With the third class mailing, I have about a week after the magazines are sent to put the companion disk together and mail it out. From now on, there will be a little disk icon (as below) if the article has items on "microdisk". Pretty much, if a file is mentioned, there will be a copy on "microdisk". I hope this isn't much of an inconvenience, it sure does take some pressure off near deadline time and avoids the embarrassment of printing the wrong thing!

Please take the time to fill out and return the survey received with this issue! After all, I keep telling you it's *your* magazine- now's the time to let me know what *you* want!

< 268'm >

Letters to the Editor

Keep up the good work! I learn something new every issue. I mostly use DECB but continue to learn more about OS-9 as I try new things. My latest effort was to build a boot disk just for the CoCoPRO! Solitaire game.

John L. Daniels
2625 Spring Place Road SE Lot #5
Cleveland, TN 37323-4800

Thanks John! I'm happy that the magazine seems to be serving you. Many readers are in your same situation-continuing to use DECB while learning OS-9. As long as there are CoCo users out there, I'm sure many will continue using DECB even when they master OS-9. There have been lots of good DECB programs over the years, and a few more still trickle in. Many don't have an equivalent in OS-9. So keep plugging away.. you'll always find some support here!

Thanks for the sample issue. Please start a subscription for me. The thing that interested me was getting some insight into the different users of OS-9. Just knowing who is using the OS and for what is useful information to me. I am hoping that your magazine might lead to contacts with people who may have goods and services that I might find helpful. I am particularly on the lookout for development tools above and beyond the normal suite of C compiler, make and the Bourne like shell that we are using (I don't know where we got it as it was acquired before I arrived on this scene). As I believe I said in my original letter to you I would love to get the source code to fix some of the bugs and incompatibilities with the real Bourne shell that drive me up a wall.

Our use of OS-9 is to provide the intelligence in a medical imaging laser printer. We are using the real time aspects of OS-9 as well as the fact that it supports Ethernet, SCSI (both for local disks and for external images) and supports our target Tadpole 68030 based VME CPU card.

We have been doing our development on the OS-9 system but are moving more and more toward our Sun based cross development environment. I'm afraid that the OS-9 development system is a poor second to the highly robust and feature full Unix environment provided by Sun.

The CoCo was my 2nd computer (though I no longer use it). My first was a 6809 system bought from someone who is long gone and whose name escapes me at the moment. It ran the FLEX operating system which I had to port and write my own disk driver for. We later bought a CoCo for games for our two sons and by reading the 68 Micro Journal that I subscribed to at the time found out that OS-9 could be had for the CoCo. The machine is the original 64K Grey case with a single disk drive that I expanded to a two drive configuration for the bargain price of $200.00. I bought OS-9 as a cheap way to run a UNIX like OS. I played with it for a spell but tended to stick with the FLEX system as I was developing a machine controller for a customer using a 6809 CPU on a board of our design and FLEX had an assembler that allowed me to write relocatable modular code which OS-9 didn't. The FLEX system still stands although I rarely use it. I find it rather amusing to have this old, floppy based 6809 system where my console terminal is a Sun SPARCstation with 16MB of memory and 2GB of disk running kermit over a serial port. I still use it because I am still supporting this same customer.

Actually I still have the original CoCo, OS-9 disks and one each of the Radio Shack an Microware OS-9 manuals. I hate throwing things out that still work.

I got involved with real UNIX later on in my work and bought these boards that I mentioned in my original letters from my employer. They were originally targeted as the main CPU for a circuit board tester running UNIX but the project was killed and I bought several boards because they would run UNIX. These boards have a 68000 CPU (8Mhz I believe) with 512K of ram, two serial ports, parallel I/O and a SASI port (SASI is the same as SCSI but doesn't support the full command set). The CPU can be upgraded to a 68010 and the memory can be upgraded to 2MB using 256K d-rams. I have complete schematics for the boards which by the way are VME compatible although not the actual VME form factor (the boards are larger). The also have a prom monitor embedded in the 27128 (4) EPROMs on board. The boards were intended to be run in a card cage but I have run them as single board computers using an external power supply. If you think that these boards could be of any use to any of your readers I would be willing to let them go at say $100.00 apiece if you think that is reasonable. I have about six of them. Let me know if it is worth advertising them.

Another thought about a platform for OS-9 is a bit off the wall but I'll mention it anyhow. The Sun 3 line uses the Motorola 68020 or 68030 chip as its engine depending upon which model 3 you are talking about. All have an FPU and all have memory management although I believe that in the case of the 020's the memory management is Sun designed. My point is that complete Sun-3's can be had for several hundred dollars on the used equipment market. I know of several vendors who would sell such systems. A typical Sun-3 has 4MB ram minimum with some expandable to 24MB using 1MB SIMMs, a built in SCSI port, a built in Ethernet port, two serial ports, video (1152x900 typically to a 19" monitor which can be either mono or color), keyboard and optical mouse. That's not a bad deal for a few hundred dollars if someone would port OS-9 to it.

Charlie Jack
jack@temerity.polaroid.com
(617)-386-4479:voice; (617)-386-4360:fax
(508)-655-1471:home

Thanks for the reply! Unfortunately, the license for OS-9/68K is a bit steep, but maybe a Sun version will come out one day...

Sources...

Every now and then I run across a great source for some product or service that might be of interest to my readers. I will include those here or in an article when I run across them, but only for exceptional deals or hard to find items. Here is one of those now...

Flat-rate classic computer repair.
CoCo 2 and 3 - $20; Floppy Controller, Disk Drive, MC-10 - $15 ; all + parts & shipping
$5 extra charge for modified equipment (anything other than factory modifications).

Computer Classics
Rt 1 Box 117
Cabool, MO 65689
417-469-4571

Send SASE with any inquiries.

I haven't tried this service, but thought it might be of interest to some. The Adam, Atari 8 bits, Timex/Sinclair, TI-99/4A, TRS-80 I-IV, Commodore 8 bit, and Osborne are also listed. Write for a quote on anything not listed, modifications done also. Turn-around time is 2-4 weeks, problem cases may take longer. He does mention that you will be notified in the event of excessive charges or delays due to unexpected problems.. BEFORE you are charged! He also has 5.25" 720K drives for $25 each, 10 for $150.

< 268'm >

Letters are printed on a space available and popular subject matter basis. If you don't want your letter printed, or wish to withhold your address and/or name, please state so when writing. In some cases, letters are edited for space and/or clarity. If a personal reply is desired, please enclose an SASE.

The Hard Drive

F.G. Swygert

History, how, and why of the most necessary peripheral in modern computing

Personal computer hard drives trace their roots to main-frame storage. In the 60s, removable platter hard drives with twelve to eighteen inch hard media were developed. These had retractable heads (so the platters could be removed) that literally flew thousandths of an inch over the surface. Rotational speed and density was better than floppy technology, but not up to modern hard drive densities due to the fact that the drives were only partially sealed. IBM solved part of this problem by developing a drive with a 60 MB capacity (in the late 60s) - 30 MB sealed and 30 MB removable. The removable portion was deemed necessary as many computer systems changed hard drive platters much as one changes a floppy- when the data on that drive is needed. This kept total drive costs down (one of these units sold for around $35,000 in 1970 dollars!). These drives were nicknamed "Winchesters" after the famous 30-30 caliber Winchester repeating rifle. Soon after, all hard drives became known as "Winchester" drives.

Floppy drives were slow (still are!)- anyone who has used (or even still uses) the old full height 5.25", 35 ms access time drives on a CoCo knows just what I mean. But do you understand WHY a floppy drive is so slow compared to a hard drive?

The biggest part of the problem is the removable nature of the floppy drive's media. Since a floppy drive is open to the world, minute particles of dust and other foreign objects invariably get on the disk and in the drive. Unless you have a class A clean room with over-pressurization and an air lock, and wear a mask and change into known clean clothes every time you enter, your computer is exposed to these contaminants. The floppy drive controller must determine if it is reading good data on your drive or a contaminant. Therefore, it can only spin up to a certain speed (300 RPMs for 5.25" 360K floppies) and it must read the data at least twice to make sure it is getting good data and not garbage (three times if there is a difference between the first two- best two out of three wins- and up to five times before an error occurs). Other factors come into play also, such as the magnetic coercivity (sensitivity) of the media, closeness of the heads to the media, etc. Since floppies have to contend with minute particles, there has to be a certain amount of space between the media and drive heads. The disks can't be to sensitive, else a simple movement in and out of the drive itself would cause damage to the data (a small magnetic field is generated even by that simple movement!).

The "Winchester" solved most of these problems by sealing the media of the fixed portion in an ultra clean environment. Since there is virtually no contamination to deal with, the platters can spin faster (usually 3,600 RPMs), heads can be closer (they literally "fly" over the platters on air currents generated by the spinning platters), and magnetic coercivity can be increased. Due to the speeds and need for absolutely smooth surfaces, platters were made of aluminum about 3/16" thick. The increased rotational speed, closeness of the heads, etc. meant much faster data transfer times, though head access time was actually a good bit slower than floppies. Why? Tracks were closer together (lack of contaminants meant data could be more densely packed). Because they were so narrow, they had to be precisely controlled. The only thing available at the time was a "stepper motor" that physically moved the head in rigid "steps". There was some amount of head bounce, so the controller had to allow for head settling. Later drives used a "voice coil" head actuator. These actuators are electromagnetically controlled and have no mechanical stops. They are therefore faster, smoother, and more reliable. No hard drives manufactured over the last two or three years use stepper motors, but there are plenty new stepper drives still on the market, especially 20 to 40 MB units.

Since the platters in a "Winchester" drive were rigid instead of flexible, they soon became known as "hard" drives. The main reason floppy drives are "floppy" is for durability- a thin aluminum platter could be bent- how would you retrieve data then? Ever try playing a warped record?. The drives were also know as "fixed disks" due to their non-removable nature.

Hard Drive Controllers

In the bad old days, connection from controller to computer was by a custom 'host adapter'. Before any system could use a hard drive, there had to be a custom host adapter AND a matching controller, plus the drive. In many cases, the host adapter and controller were built as a unit (such as with IBM "controller" cards). In others, there was a host adapter that had cables between it and the controller, which was usually housed in the drive cabinet (if the cabinet was separate), and another set of cables between the controller and drive. This is how Tandy set up the only factory hard drive unit for the CoCo. A host adapter plugged into the Multi-Pak, then cables went to a standard Model I-IV type external hard drive subsystem (case, power supply, controller, and 10 or 15 MB drive). The Model I-IV systems plugged directly to the computer (expansion chassis for Model I). The CoCo host adapter converted between the CoCo and Model I-IV busses.

Seagate developed the ST506/412 standards used by MFM and RLL format drives with the inadvertent help of IBM. The ST506 was a modification of the SASI standard for a particular Seagate drive- a 5.25", full height, 60 MB drive. The ST412 standard added buffered seeks, but was otherwise identical to the ST506. What made this a "standard" was it's adoption by IBM for the PC/XT in 1983. Rather than develop different standards, hard drive manufacturers adopted the Seagate interface (which was royalty free). Many older SCSI systems actually use a separate controller card which accepted SCSI on one end and out-put SASI on the other so that a low-cost drive could be used. Newer SCSI drives have the controller 'embedded' right on the drive and accepts SCSI signals directly.

Types of Hard Drives

There are five common types of hard drives: MFM, RLL, SCSI, ESDI, and IDE. MFM and RLL refer to the data encoding scheme of ST506/412 interface drives. The other three refer to the interface type.

MFM (Modified Frequency Modulation) This is the same data encoding scheme as used by floppy drives. The magnetic pattern placed on a drive by an MFM drive are evenly spaced over the platter surface. Single bit errors are easily detected under this format. Under the ST506/412 standard, all MFM drives have 17 sectors per track. Due to the nature of the MFM format, only so much data can be stored in a given space. Therefore, physical size of the dive limits storage capacity.

RLL (Run Length Limited) RLL encoding greatly increases the density of data (usually by 50%- a 20 MB MFM drive would store 30 MB with an RLL format) and the data transfer rate of a hard drive. Although not generally published, there are two numbers associated with RLL drives- the minimum run of zeroes and the maximum number of zeroes (or limit) between two ones, usually 2 and 7 respectively. Since the intervals between data bits are not regular, timing between bits by the controller is of utmost importance. Since the encoding used is not fixed in length, a single bit error cannot be detected and may corrupt as many as five bits. Due to the increased density, a drive must be certified as RLL compliant. Although some MFM drives will retain information when formatted with an RLL controller, it isn't recommended that one try. The information may be there a few days or a few months- then suddenly become unreadable. The quality of the platters of RLL certified drives is a bit higher than standard MFM drives out of necessity. RLL signals are more sensitive to interference than MFM signals. Cable lengths should be kept to a minimum and routed away from power supplies, fans, and other noise inducing devices. If used with a CoCo, shielded cables would be advised. Stepper-motor drives have to many temperature and tracking related errors to be reliably used with RLL encoding, so avoid using RLL controllers with such drives even if they are RLL certified. An RLL drive can be used with a standard MFM controller, but will have a 33% reduction in capacity.

SCSI (Small Computer Systems Interface- pronounced "scuzzy") Attempting to form some sort of standard for the adapter/controller connection, the SASI (Shugart Associates Standard Interface, designed by- you guessed it- Shugart Associates) and later SCSI (Small Computer System Interface - a sort of generic, improved version of SASI) interfaces were formed. SCSI is not a hard drive interface specifically but a general purpose interface bus. The SCSI system consists of a host adapter (sometimes integrated into a computer, such as on the Apple Macintosh and Hazelwood 68K boards) and up to seven devices. Theoretically, one could have fourteen hard drives connected to a single SCSI port (seven hard drive controllers supporting two drives each). Most SCSI specific hard drives have a built-in controller, which reduces this theoretical limit to seven hard drives. Built-in controllers (commonly referred to as "embedded controllers") usually use an RLL encoding scheme. Since the controller is right on the drive, interference and cable lengths are moot points. The SCSI bus is buffered and more durable than the RLL signals generated on the drive itself. When using an embedded controller drive, the encoding scheme used by the drive itself is unimportant, as only the SCSI communications protocol can be used to communicate with the drive.

ESDI (Enhanced Small Device Interface) This is a specialized hard/tape drive interface established in 1983 by Maxtor corporation. ESDI is similar to SCSI in that it uses embedded controllers. This allows the number of sectors per track to be doubled over the ST506/412 standard of 17 (to 34 sectors/track) and the data transfer rate to be increased as well. These drives have the ability to read the drive's capacity parameters directly from the drive and control error mapping without user input. For a short time, from around 1986 until about 1990, ESDI was very popular with 70 MB or larger drives.

IDE (Integrated Drive Electronics) These are the newest type drives, developed over the past four to five years. Basically, IDE integrates all the drive electronics on the drive itself, eliminating the need for a controller card. The signals required by an IDE drive are virtually the same as the standard PC/AT class sixteen bit expansion bus. Only five of the signals required are not generated by the expansion bus itself, and a few of the signals are inverted. What this means is that five inexpensive, "glue logic" ships are all that is needed to convert between the PC/AT buss and an IDE drive. Except for those five extra signals, one could plug a cable directly into the expansion connector! Because only five simple chips are needed, many manufacturers adopted IDE early on. It was a simple, cheap matter to integrate the "controller" (actually just a host adapter) on the motherboard. Some manufacturers reduced the chip count to one or two by producing a special chip or programming PALs or GALs for the new interface. IDE systems are also "smart"- the embedded controller takes care of nearly every drive related function. The only OS-9 based system currently capable of supporting IDE drives use Peripheral Technologies or Computer Design Services motherboards (the Delmar System IV and System V). A more detailed article on IDE will be printed in a future issue.

What Made Hard Drives Necessary

Anyone who works with MS-DOS or even Unix based computers can easily understand why hard drives have become absolute necessities for those systems. For starters, the operating systems themselves have become enormous. MS-DOS version 5.0 requires at least 1.5 MB of hard drive space (without DOSSHELL). Programs are exceptionally large- Windows requires a minimum of 8 MB of hard drive space, a program such as Microsoft's Word for Windows at least 6 MB Running these from floppy drives would be impossible!

Another advantage to the hard drive is speed. Even an old drive with a 65 ms access time is faster than the quickest floppy drive. The data transfer rate is about ten times faster than a modern floppy. This means that commands no longer have to be loaded into memory to be accessed quickly- operating systems and programs can be made larger and more versatile since only small portions need be memory resident. OS 9/OSK really shines with a hard drive, as all commands are almost instantly available and very few need to be loaded into precious memory.

Summary

Well, now you know all about hard drives- the history, common types, and why they are deemed necessary in today's computing environments. Is a hard drive in your future? If you own an OSK machine, it most likely is already, or soon will be (OSK will run from a 720K floppy). Armed with this information, one should be ready to determine what type of drive is desired, or at least ready to ask the right questions before a purchase is made.

< 268'm >

CoCo Hard Drives

F.G. Swygert

Ken-Ton SCSI Interface- up to 7 hard drives for the CoCo!

The Ken-Ton Electronics (KT) SCSI hard drive system for the CoCo is the only true SCSI controller available. Coupled with RGB-DOS in ROM, it is truly unbeatable!

As this is being written, I just got off the phone with Mr. Joe Scinta (pronounce that "shinta"). The reason he has not been making much effort to sell these systems is tech support. Basically, anyone who buys a complete KT system, drive and all, will hardly ever have a problem. The system is that foolproof when shipped from the factory whether one chooses to use DECB or OS-9. The problems occur when people buy systems and piece them together themselves, or decide they want to make drastic changes in the way the system operates. When the system set-up has been damaged to the point the drive won't respond, a couple hours on the telephone is probably in order- time that can't be made up, and costs KT $30 an hour. Mr. Scinta runs an electronics business full time- they make all kinds of boards for any and everyone, one to a thousand or more. Unlike many vendors, this is not a "hobby" or part time venture. The CoCo support side, however, pretty much is, as there was never much profit in selling CoCo hard drives. Why did he even do it? Simply a labor of love for the CoCo, and the realization that CoCo users really could benefit from a hard drive. In fact, if a CoCo hadn't been used at Ken-Ton for many years during the business' early days, there may have never been a KT SCSI controller.

Back to the controller itself! The KT board is a true SCSI adapter for the CoCo. It is designed to run from a simple, non-powered Y-cable. The CoCo 3 supplies plenty of current for the low-power KT, a floppy controller, and even an RS-232 Pak. Being a true SCSI adapter, up to seven devices are easily supported. The board boasts a true open collector design and cannot harm any hard drive. After all, Mr. Scinta is an electronics engineer!

RGB-DOS

RGB-DOS really makes the KT stand out. This system is really just an enhancement of DECB, adding some commands and patching others to allow hard drive operation. Once burned into a ROM and installed in the KT controller board, nothing can be simpler to operate. As soon as the CoCo is turned on, it comes up in the RGB-DOS menuing system. If you want to go directly to OS-9, simply choose OS-9 from the menu by pressing "O". OS-9 will then boot from the hard drive. Do you need more than one boot for OS-9? Then install a different boot on however many DECB type drives as you need, then choose the boot desired from the menu. What could be simpler? Personally, I prefer this methodology over the B&B XTROM which auto boots OS-9, as RGB-DOS allows multiple boots.

This is probably the best hard drive system for DECB. Once the drive is formatted, you tell RGB-DOS to establish up to 251 156K areas (the capacity of a 35 track single sided drive) and number them as drives 4 to 255. The first four numbers are reserved for the standard floppy drives. The only reason OS-9 operations was mentioned in this article first is that most people shopping for a hard drive are doing so for OS-9. But with RGB-DOS , there is no reason to have OS-9 at all. The hard drive functions flawlessly under DECB alone and will totally utilize a 40 MB hard drive. Even many machine language programs that won't operate from other hard drive systems will function under RGB-DOS. For those few that won't, a command is provided that switches the first four drive letters to four of the other hard drive residing areas.

DECB and OS-9 on the same hard drive is easy. Simply use the provided utility to format the hard drive under DECB (it is easier to do under DECB than OS-9) and then "partition" the drive. What RGB-DOS actually does is designate so many tracks for DECB use and the remainder for OS-9, it doesn't really partition the drive in the actual sense of the term. After that one simply loads the hard drive with DECB software and the OS-9 operating system and software.

The KT controller doesn't work well with no-halt floppy controllers on a Y-cable, or the Disto Super Controller I. Use with a Multi-Pak and possibly a Slot-Pak should be no problem. According to Mr. Scinta, the problem lies in timing. The other controllers simply weren't designed to operate from a Y-cable like the KT, and the added capacitance incurred with a Y-cable throws timing signals off enough to affect the entire computer system. Any of the short controllers, including the Disto mini, work fine from a Y-cable or MPI. In actual use, the only time a halt will occur is when accessing the floppy drives, since the KT interface doesn't use interrupts.

Speed!

I have heard for a long time now that the Burke & Burke CoCoXT was the fastest hard drive system available for the CoCo. Mr. Scinta disputes this. He has run test on the CoCoXT, Disto, and FHL Eliminator systems against the KT. The Disto and B&B transfer 1 MB of data in 80-100 seconds, the Eliminator in about 40 seconds, and the KT in 26 seconds flat. The reason for this is that the KT supports true hardware hand shaking between the drive and interface- nothing is done in software. The only limiting factor in transfer speed is actually the CoCo itself- it simply can't run any faster! With the B&B and Disto, a lot depends on the speed of the drive itself.

What Type Drive?

The KT interface will actually work with any SCSI drive. If the drive doesn't support 256 byte sectors, one will loose about half the capacity. With a 512 byte sector drive, RGB-DOS simply ignores the extra 256 bytes on each sector. If one's primary reason for buying a hard

drive is for OS-9, this isn't as big a problem as it appears. Mat Thompson's (available from Northern Xposure) 512 byte sector drivers for OS-9 work wonderfully with the KT. So the only lost capacity would be in the DECB section. Even if one is primarily using the hard drive for OS-9, it would be advisable to set up at least a half-dozen or so DECB drives for multiple boots and those one or two DECB programs that are often used. Setting up 12 DECB drives would mean losing only 3072 bytes- 3K!

All factory KT systems ship with Seagate "N" series drives, the type recommended by Mr. Scinta. All Seagate "N" series made after 1990 support 256 byte sectors so no space is lost. Refurbished drives and those made prior to 1990 may not support 256 bytes sectors, even if the refurbs carry a 1990 or later date tag. A refurbished drive may have an older circuit board on it. Some other drives also support 256 byte sectors. If the primary use is DECB, a 256 byte sector drive is almost a must, so don't buy another model or refurbished Seagate "N" unless ti can be returned if it doesn't format at 256 bytes per sector.

Floptical CoCo Drives? YES!!

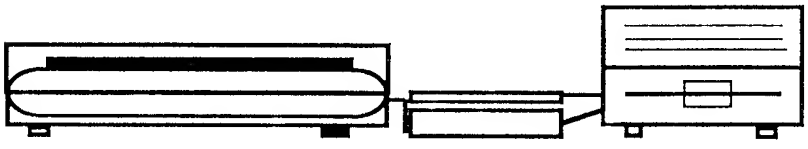
Since the KT interface is a true SCSI device, any SCSI device can be used. With this in mind, Mr. Scinta connected a floptical drive to his CoCo many years back. The driver was easy, since the floptical works very much like a standard floppy or hard drive. Only a few commands needed to be added. If there is much interest, the driver will be made available to the general public. Currently, 20 MB floptical drives are around $300 for the drive and $20-$25 for disks. It is possible to connect ANY SCSI device to the KT, but no drivers have been written to support tape drives or other devices, but an enterprising programmer could write the necessary drivers. If someone does, let me know!

Availability

I originally called Ken-Ton electronics because I got a disturbing message that they were turning drive requests away. I've already mentioned the problem- technical support was just to expensive. Mr. Scinta is still interested in supporting CoCo users even though there is little profit in it for him. Our lengthy conversation ended up with the fact that he is more than willing to supply systems and parts to someone willing to provide technical support, sales, and advertising. With that in mind, FARNA Systems ended up becoming the source for Ken-Ton hard drive systems by the end of the conversation. We will be with Mr. Scinta at the Chicago CoCoFest showing the systems, and several will be available at yet to be determined show prices. The normal price will be $610.00 (shipping and insurance included for UPS ground) for a complete, ready to run 85 MB system. This includes a power supply, case, KT interface, Y-cable, RGB-DOS in ROM, OS-9 drivers, and a brand new two year warranty Seagate drive (5.25" half height). At this time, individual components are not being offered, but there will be a "no-drive" kit available after the Chicago CoCoFest. The no-drive kit will include all but the drive, case, and power supply. It too will be almost ready to run, and will include RGB-DOS already in ROM.

Coming up next...

There are two other hard drive systems currently in use for the Color Computer. The most widely known is the **Burke & Burke CoCoXT**, which will be discussed in detail in the next issue. Following that, we will discuss the **Disto SASI/SCSI** controllers. Both are excellent systems that have seen years of CoCo service. *< 268'm >*



Ken-Ton System with hard and floppy drives in same case. KT interface is on top of floppy controller, both on a short Y-cable. The KT interface is on of the few CoCo peripherals designed specifically for use with a Y-cable.

OS-9/68000 Hard Drives

F.G. Swygert

Hard drives for Delmar Co. Systems (PT68K) and FHL KiX (Hazelwood)

A hard drive is nearly a necessity with an OS-9/68K (OSK) system. This magazine supports three main OSK systems- those based on Peripheral Technologies and Computer Design Services 68000 and 68020 boards that use ISA expansion busses (8 and 16 bit, respectively), and Hazelwood 68020 and 68030 boards that uses a 32 bit Euro-K bus.

FHL Kix 20 and 30 (Hazelwood 68020 and 68030 boards): These systems use a common SCSI 512 byte sector hard drive, even SCSI II drives will work. There isn't a whole lot to say- these are available from any PC supplier. In recent years, SCSI has become popular among PC users, so surplus and refurbished drives are often available.

Some really good deals can be found on small and large SCSI drives, but not much in between (see "HD Sources"). The smal (20-70MB) drives just aren't considered big enough any more. Large drives (500MB-up) have always been popular in SCSI because the ST506/412 interface is a bit to slow for large drives.

Delmar System IV and System V (Peripheral Technologies PT68K series and Computer Design Services CD68X20 boards): The PT68K and CD68X20 boards were both designed by Frederick Brown of Atlanta, GA. They use a common ISA bus for expansion cards. There is one major difference between the boards- the PT68K series uses an eight bit (PC/XT) expansion bus while the CDS uses a sixteen bit (PC/AT) expansion bus.

The CD68X20 has an IDE and SCSI port built onto the motherboard. The IDE port is the choice for a hard drive. IDE drives are very common now in sizes up to 1.6GB (giga-bytes; 1,000MB). The SCSI port is useable with a hard drive, but drivers are not included with OS-9 (extra cost option). Fred Brown doesn't recommend using the SCSI port, mainly because IDE drives are faster than most SCSI types (up to 1.6MB per second).

The older PT68K boards require an eight bit PC/XT type card, either MFM or RLL types. The WD-XTGEN or 1002 PC cards are generally preferred, although most generic MFM and RLL cards should work just fine with the supplied drivers.

An IDE drive may also be used with the PT68K. The eight bit IDE card is basically a sixteen to eight bit adapter with the few extra control lines required by the IDE drive. There are only two or three manufacturers of eight bit IDE cards. Peripheral Technologies only supports one of them. A card can be ordered from PT with the required drivers for $69. The card alone from most PC suppliers is around $50, so this is an excellent price for the card and drivers.

< 268'm >

HARD DRIVE SOURCES

After a search through various sources, these came up as the best deals . Happy hunting!

Midwest Electronics
124 12th Avenue South
Minneapolis, MN 55415
612-339-9533

20MB MFM $40
40MB MFM $69
Other items of interest:
CGA Monitors $99
VGA Monitors ..... $170 & up
Epson 3000 24 pin printer .... $ 149

Prime Components, Inc.
33 Freeman St.
Newark, NJ 07105
201-344-1029

New ST225 20MB MFM,5.25" $34.95
(limited quantities- hurry on this one!)
Other items of interest:
Victor/Honeywell 101key AT/XT switchable keyboards ..... $24.95 (6 pin PS/2 connector)

Alltech Electronics Co.
602 Garrison Street
Oceanside, CA 92054
619-721-7733

Openface SCSI HD case and P/S (5.25" half high) single.....$45.00 double.....$65.00

Tredex
800-899-6800

Refurb CGA Monitors.....$39.99
(okay, so it's not a hard drive...GREAT deal!)

Crazy Bob's
50 New Salem Street
Wakefield, MA 01880
617-246-6767

Mixed 35MB MFM, 5.25" $49.00
Quantum Q280 80MB SCSI, 5.25..... $89.00
Quantum P80S, 80MB SCSI, 3.5" $99.00
Conner 212, 200MB SCSI, 3.5" $199.00
All half-high except 200MB (1" high). No brand choice for 35MB drives.
Other items of interest:
360K half high 5.25" floppy drives.....$9.00
720K 3.5" floppy drives.....$39.00
1.44MB 3.5" floppy drives.....$49.00
"Universal" 8 bit MFM HD only controller..$39.00

Computer Disk Service
3537 Old Conejo Road #112
Newbury Park, CA 91320
805-499-6355

Miniscribe3425SA 21MB SCSI, 3.5" .. $35.00
Call for other specials.

Micro Machines
2120 Howell Avenue, Ste. 404
Anaheim, CA 92806
714-978-2220

ST4096 80MB MFM, 5.25".....$149.00
ST506 75MB MFM, 5.25" $119.00
ST277N 65MB SCSI 5.25" $139.00
ST157N 42MB SCSI 3.5" $99.00

Hi-Tech Component Distributors, Inc.
59 S. LaPatera Lane
Goleta, CA 93117
805-681-9961 (ask for drive sales)

ST1400N 327MB SCSI, 3.5".....$299.00
Fujitsu M2261HA 357MB SCSI, 5.25".....$259.00
ST4702N 613MB SCSI, 5.25" (FH).....$449.00
ST41600N 1.3GB SCSI, 5.25" (FH).....$949.00
ST4053 full height 42MB MFM.....$69.00
ST251 half high 42MB MFM.....$89.00
Conner CP3040 40MB SCSI, 3.5".....$99.00
Generic MFM 8 bit HD only controller.....$14.00
Generic RLL 8 bit HD only controller.....$21.00
Floppy Drives - 360K.....$12 720K (3.5").....$15
1.2M.....$32 1.44M.....$29

JEM Computers, Inc.
617-497-2500

ST325X 21MB MFM, 3.5".....$29.00
ST4051 42MB MFM, 5.25" (FH).....$79.00
Quantum Q280 76MB SCSI, 3.5".....$99.00
CDC 94161-155 141MB SCSI, 5.25" (FH)..$119.00
ST41200NM 1.03GB SCSI, 5.25" (FH).....$699.00

The HD6309E CPU

Robert Gault

PART 2: Let's use some of the new 609 features to speed CoCo up (DECB)!

We are going to start replacing critical sections of the Basic ROM code with 6309 routines in an effort to speed up Coco3 functions. Owners of older Coco 1 or 2 units can also benefit but the greatest effect will be seen in the Coco3 high resolution 40 and 80 column text modes and high res graphics.

To make changes in the Coco 1 or 2 systems it is necessary to be in the 64K RAM mode. This requires running a utility to copy the ROMs to RAM and set the computer to run in that mode (see ROMRAM on microdisk). The Coco3 is already in full RAM mode in its normal state. There is a problem, however, with both the Coco1 & 2 as well as the Coco3 when the reset button is pressed; any changes made to the system disappear. So the first thing we need to do is find out how to protect our changes from the reset button.

In order to accomplish this and any other ROM patching task, we must know the ROM code; no small task. There is only one good source for this information and that is the "Color Basic Unravelled" series by Spectral Associates. I don't know if it is still available from any source but the information contained is indispenable. I will be referring to these disassemblies of the Coco ROMs on occasion.

It will not be possible to fully explain all of the code which will follow in this or subsequent articles. The reader is refered to the above series or other disassemblies of the Coco Basic for full understanding.

Reset Protection

The reset button forces the Coco into the mixed RAM/ROM mode; $0-$7FFF RAM and $8000-$FF00 ROM. When this happens all new code above $8000 will disappear, so our reset protection code must be in low RAM. There is very little space available for this purpose. The most obvious choices, the cassette buffer, and extra bytes from $149-$151 are almost always used by a program or third party DOS.

My own personal choices are the top of the random file buffer area just below $E00 and the top of the user command vector table just below $973. Few Basic programmers make use of all ten user commands or the entire file buffer area. The initial short section of code below is located in the user command area.

```
00100 * Coco RESET protection using
6309 code.
00110
00120 lngcde equ 7 length of
initial RESET protection code
00130 ramode equ $ffdf a store to
this register engages RAM mode
00140 what equ $b9af print "?" to
consol
00150 diskwm equ $c0e7 Disk Basic
warm start routine
00160 palet equ $e678 Coco3
custom palette images
00170 palset equ $e634 Coco3
palette setting routine
00180 clock equ $ffd9 a store to
this register starts 2MHz CPU clock
00190 coldst equ $fffe vector to the
cold start routine; any Coco
00200
00210
00220 org $72 warm start
vector
00230 fdb reset
00240
00250 org $973-lngcde
00260 reset nop manditory flag
00270 sta ramode start RAM
mode
00280 jmp morcde jump to rest
of REST protection code
00290
00300 * The next origin must be in an
unused protion of memory. Typical
00310 * choices are copyright messages
and the DLOAD code area at $8d14-$8e37
00320 org $e333 Coco3
copyright message area
00330 morcde sta clock reset CPU
clock to user desired rate.
00340 oim #1 $ff03 engage
vsync interrupt
00350 ldx #palet point to custom
palette colors
00360 jsr palset reset the palette
registers; Coco3
00370 eim #$12 diskwm test for
valid Disk Basic warm start
00380 lbeq diskwm
00390 jmp [coldst] no Disk Basic
so do a cold start
00400
00410 end what attempts to
execute will print "?"
```

```
00100 * Coco RESET protection using
6809 code.
00110
00120 lngcde equ 7 length of
initial RESET protection code
00130 ramode equ $ffdf a store to
this register engages RAM mode
00140 what equ $b9af print "?" to
consol
00150 diskwm equ $c0e7 Disk Basic
warm start routine
00160 palet equ $e678 Coco3
custom palette images
00170 palset equ $e634 Coco3
palette setting routine
00180 clock equ $ffd9 a store to
this register starts 2MHz CPU clock
00190 coldst equ $fffe vector to the
cold start routine; any Coco
00200 nop equ $12 value of
opcode NOP
00210
00220
00230 org $72 warm start
vector
00240 fdb reset
00250
00260 org $973-lngcde
00270 reset nop manditory flag
00280 sta ramode start RAM
mode
00290 jmp morcde jump to rest
of REST protection code
00300
00310 * The next origin must be in an
unused protion of memory. Typical
00320 * choices are copyright messages
and the DLOAD code area at $8d14-$8e37
00330 org $e333 Coco3
copyright message area
00340 morcde sta clock reset CPU
clock to user desired rate.
00350 lda $ff03 get PIA setting
00360 ora #1 engage vsync
bit
00370 sta $ff03 tell PIA
00380 ldx #palet point to custom
palette colors
00390 jsr palset reset the palette
registers; Coco3
00400 lda diskwm read first byte
of Disk Basic warm start code
00410 cmpa #nop is manditory
flag present?
00420 lbeq diskwm
00430 jmp [coldst] no Disk Basic
so do a cold start
00440
00450 end what attempts to
execute will print "?"
```

Note the two 6309 instructions OIM "OR immediate with memory" and EIM "exclusive OR immediate with memory". Use of these instructions is significant above not for a speed increase but for a noticable shortening of code. Also note how the EIM was used as a compare (CMP) instruction. Use the above code by assembling it and LOADMing it into memory. The OIM instruction above is fcb $71,1,$ff,3; the EIM is

fcb $75,$12,$ff,3 for readers with a 6309 but not a 6309 ready assembler.

The following Basic program will poke into memory the required 6309 code.

```
10 REM RESET PROTECTION FOR
COCO3
20 LI=90:START=&HE333:FINISH=
&HE34B:GOSUB50
30 LI=120:START=&H96C:FINISH=
&H972:GOSUB50
40 POKE&H72,9:POKE&H73,&H6C:
END
50 FOR M=START TO FINISH STEP 10
:SUM=0
60 FOR I=0TO9:READ A$:VA=VAL
("&H"+A$):SUM= SUM + VA:POKE
M+I,VA:NEXT:READ CHK: IF SUM<>
CHK THEN PRINT"ERROR IN LINE"
LI: END
70 LI=LI+10:NEXT:RETURN
90 DATA B7, FF, D9, 71, 1 , FF, 3 , 8E,
E6, 78, 1519
100 DATA BD, E6, 34, 75, 12, C0, E7, 10,
27, DD, 1305
110 DATA 9F, 6E, 9F, FF, FE, 00, 00, 00,
00, 00, 937
120 DATA 12, B7, FF, DF, 7E, E3, 33, 00,
00, 00, 1083
```

Now that we can be sure that are code patches won't disappear on RESET, let's look at a patch to the Coco3 high resolution text screen scroll routine. That is the routine that moves your text up the screen and one of the factors in making 80 column LISTings so much slower than 32 column LISTings. Below is the original code.

```
  org  $f854
  pshs  d
  ldx  #hresscrn point to start of hi-res text
screen
  lda  h.column  get characters per row
  cmpa  #40
  bne  $f86e
  * scroll a 40 width screen
  $f860  ldd  2*40,x  get one character
+ attribute in next row
    std  ,x++  move it up one row
    cmpx  #hresscrn+(rowmax-1)*40*2
end of screen?
    bcs  $f860  no, then keep going
  $f86a  bsr  $f87b  fill last row with
spaces
    puls  d,pc  back to calling routine
  * scroll an 80 width screen
  $f86e  ldd  2*80,x
   std  ,x++
    cmpx  #hresscrn+(romax-1)*80*2
    bcs  $f86e
    bra  $f86a
```

The above code is quite reasonable. The limiting factor is the loop speed for the ldd std cmpx bcs instructions. The only way to speed this up is by using the new functions of the 6309 CPU.

There are two approaches that could be used: 1) replace the 2 byte register D with the 4 byte register Q; 2) use the block transfer functions. Approach #1 will cut the loop time about in half as twice the information is transferred per loop but would take more bytes of code than the original. Approach #2 will be faster still as it will take only 3 clock cycles to transfer a byte instead of 11.5 in the above code; about 4 times faster. It will also take less code than the original.

```
00010  title  A 6309 patch for Hscroll
00020
00100  org  $f854
00110  pshs  d,y  save an extra register
00120
00130  org  $f860  40 width scroll
00140  ldw  #40*2*23  number of bytes
to move
00150  leay  40*2,x  point to next row
down
00160  tfrp  y,x  transfer bytes by
indexed from y to x
00170 *  and update both y and x by 1
00180
00190  org  $f86c
00200  puls  d,y,pc  remember to
recover extra register
00210
00220  org  $f86e  80 width scroll
00230  ldw  #80*2*23  number of
bytes to move
00240  leay  80*2,x  point to next line
00250  tfrp  y,x  transfer from y to x
and increase by 1
00260
00270  end  $b9af  do nothing on
EXEC
```

tfrp is the EDTASM6309 memnonic for tfr (r1+,r2+) or tfm+. There is no approved memnonic for this instruction and I opted for simplicity. Code is fcb $11,$38,$21 for tfrp y,x.

Assemble the above, load it into memory, and the change will be easily apparent when any program is listed. The same type of patch could be made to the 32 column screen but that scroll is already too fast for easy reading.

The following Basic program will poke the required 6309 code into memory.

```
10 REM SCROLL ENHANCER FOR
COCO3 40 & 80 COLUMN SCREENS
20 LI=80:START=&HF860:FINISH=
&HF869:GOSUB30
22 LI=110:START=&HF86E:FINISH=
&HF878:GOSUB30
24 POKE&HF854,&H34:POK E&HF855,
&H26:POKE&HF86C,&H 35: POKE
&HF86D, &HA6: END
30 FOR M=START TO FINISH STEP 10:
SUM=0
40 FOR I=0TO9:READ A$: VA=VAL
("&H"+A$): SUM= SUM+VA:POKE
M+I, VA: NEXT: READ CHK: IFSUM<>
CHK THEN PRINT"ERROR IN LINE"
LI: END
50 LI=LI+10:NEXT:RETURN
80 DATA 10, 86, 7 , 30, 31, 88, 50, 11, 38,
21, 576
110 DATA 10, 86, 0E, 60, 31, 89, 00, A0,
11, 38, 679
120 DATA 21, 00, 00, 00, 00, 00, 00, 00, 00,
00, 33
```

< 268'm >

Programming in C

(continued from page 23)

REMEMBER: If x[i]+j is a pointer, then *(x[i]+j) is what it points to... and that means that *(x[2]+3) and x[2][3] are EQUIVALENT

What's in a name ? (of an array)

The name of a 1-dimensional array is a pointer to the first element of the array! So, if x is an array, declared as int x[10], then x is a pointer to x[0] (since x is 1-dimensional). And, if x is an array, declared as int x[10][15], then x[7] is a pointer to x[7][0] (since x[7] is 1-dimensional). And finally, we may printf ("%d",x[7][2]); - to print the value of the element, OR we may printf ("%d", *(x[7]+2));.

That's all for this lesson! Next issue: More Functions

au revoir!

P.J.Ponzo
Dept. of Applied Math
Univ. of Waterloo
Ontario N2L 3G1

< 268'm >

DON'T LET YOUR SUBSCRIPTION EXPIRE!

The last issue on your current subscription is on your mailing label after your name. If "6/94" is after your name, then the 15 June 1994 issue will be your last.

The Hardware Hacker

Dr. Marty Goodman

Odds 'n ends for every CoCo owner...

This column will consist of a number of unrelated, but hopefully useful tips for those who are tinkering with the hardware of a Color Computer 3, and for those interested in serial data exchange between a Macintosh RS-422 port and another computer (such as a CoCo, MM/1, or PC compatible) that uses a RS-232 port.

Color Computer 3 Addressing tip:

I've mentioned before that the Color Computer 3 decodes the addresses of its two PIA chips a bit more completely than does the Color Computer 1 or 2. Specifically, I've noted that the address for the PIA chips on the CoCo 3 is decoded using pin 13 of the 74LS138 chip on the mother board, which is active low for the address ranges of $FF00 thru $FF1F and $FF20 thru $FF2F. The A5 line is then used either inverted or non-inverted to select which of the two PIAs is being addressed. Thus, each PIA's group of four valid addresses "ghosts" four times. That is, if you can address the data port of the PIA at $FF00 at address $FF00, $FF04, $FF08, or $FF0C. The same is true for the next three registers in that PIA, and the same sort of thing holds for the four registers in the second PIA chip that "officially" are at $FF20 thru $FF23. On the CoCo 1 and 2, the PIA registers "ghost" thru not four, but EIGHT duplications of their addresses, so that (for example) the register of the PIA at $FF00 is duplicated at $FF04, $FF08, $FF0C, $FF10, $FF14, $FF18, and $FF1C. The significance of this is one has "free" address space on the CoCo 3 at $FF10 thru $FF1F and at $FF30 thru $FF3F that can be used for addressing other hardware devices one constructs. All this I've noted before.

Now, here's some new information: IF you want to address a chip to the CoCo 3 bus, Here's a new and different and sneaky way to do it without yourself fully decoding the address bus. Look at pin 13 of the LS138 chip on the CoCo 3. This is UNUSED, normally. Examination of this pin using a logic probe and some test software reveals that it is active low in the following address ranges: $FF10 thru $FF1F, $FF30 thru $FF3F, AND $FF60 thru $FF7F. Thus, you can use this pin to assist you with decoding hardware devices for the Color Computer 3 that you plan on mounting inside the CoCo 3, or (if you bring this pin out to the system bus somehow) you can use it for cards you plug into the CoCo. If you are using a RS-232 pak in your system, you will have to gate this pin 13 line with an inverted A6 line to eliminate the $FF60 thru $FF7F range. Still, using a single small scale logic chip (a 74LS02 quad NOR gate would do nicely here) and messing with only that one A6 address line in addition to the line from pin 13 of the LS138, you can address a device to the $FF10-$FF1F and $FF30-$FF3F range, putting it out of the way of anything else in your CoCo's hardware. If you use the A5 line too, you can narrow that down to one of those two ranges. Note that pin 13 appears to be valid for both read and write operations, as opposed to, say, the internal and external ROM decoding lines that come off the same chip (pins 14 and 15) which are valid ONLY during READ cycles.

Macintosh Null Modem Cables:

Macintosh computers use RS-422, not RS-232 These use two lines for each data signal. This allows RS-422 to be more rugged, and be transmitted over considerably longer cables than RS-232 It facilitates higher baud rates, too. However, it makes for some confusion when you are trying to make up a null modem cable to go between a Macintosh and a computer that has a RS-232 port on it.

Once you've figured out what sort of RS-422 connector your Macintosh has, and what its pin out is, the key to making up such a cable is this: On the RS-422 port:

(1) Consider the RS-422 RxD- to be the same as the RS-232 RxD line.

(2) GROUND the RS-422 RxD+ line .

(3) Consider the RS-422 TxD- line to be the same as the RS-232 TxD line.

(4) Ignore the RS-422 TxD+ line.

IF making up a "full handshaking" null modem cable:

(5) Hook the Handshake Input of the RS-422 port to the DTR line of your RS-232 port.

(6) Hook the Handshake Output of the RS-422 port to the RS-232 CTS line.

IF making up a "no handshaking" null modem cable:

(5a) connect the Handshake Input to the Handshake Output (if you

have one on your RS-422 port) OR to a source of +5 to +12 volts (if you don't have a Handshake Output line on your RS-422 port).

Now, old Mac pin out (DB 9) is as follows:

1 frame ground
2 +5V
3 signal ground
4 TxD+
5 TxD- (to RxD of RS-232)
6 +12V (not present on some versions!)
7 handshake (to DTR of RS-232)
8 RxD+ (ground)
9 RxD- (to TxD of RS-232)

The pinout of the 8 pin mini DIN on modern Macintosh computers is:

1 handshake Output (to RS-232 CTS)
2 handshake Input (to RS-232 DTR)
3 TxD- (to RxD of other computer)
4 frame ground / signal ground
5 RxD- (to TxD of other computer)
6 TxD+
8 RxD+

() = suggested used in handshaking null modem cable

Here's an "ASCII graphic" of the Macintosh 8 pin mini DIN to help you identify pins by number:

```
      *
   6  7  8
   3  4  5
  * 1  2 *
```

(* indicates NOTCHES in the connector)

Just for your convenience, here's the pin out for normal modem RS-232 connectors used on PC compatibles, OSK systems, CoCo Deluxe RS-232 paks, etc.

Pin Function	DB9 type connector	DB25 type connector	direction
CD carrier detect	1	8	to computer
RxD receive data	2	3	to computer
TxD transmit data	3	2	from computer
DTR data term ready	4	20	from computer
signal ground	5	7	(NA)
DSR	6	6	to computer
RTS request to send	7	4	from computer
CTS clear to send	8	5	to computer
RI ring indicator	9	22	from computer
Frame Ground	(NA)	1	(NA)

DMP 13x printer cable tip:

If you are trying to hook a DMP 130, 130a, 131, 132, or 133 printer to a PC compatible, and find that the moment you plug in the PC compatible printer cable to the printer (when the other end of same cable is already attached to your PC compatible) the printer "locks up", the cause is that Tandy, in its infinite perversity, chose to put the *INIT line of those printers on a NON STANDARD PIN (!!!). Virtually ALL other printers with 36 pin "Centronics Style" parallel ports have their *INIT line on pin 31 of that port. And either don't use pin 33, or have that pin grounded. TANDY put the *INIT line on pin 33 of that series of printers. Now, most PC printer cables GROUND pin 33 of that cable. The result: When you hook one of those printers to a PC compatible with a standard cable, it continuously forces pin 33 to ground, keeping the printer in a permanent INIT (reset) state. The fix is relatively simple: Get

a printer cable that has a 36 pin connector on it of a sort you can open up (not the molded plastic kind!). Then desolder the wires that go to pins 33 and 31 of the 36 pin connector on that cable, and SWITCH them. Then put a lable on that cable: "For Use With Tandy DMP 13X type printers ONLY!". A cable so-prepared will allow use of the DMP 13X printers with PC compatible computers.

(*editor:* There are two "quick fixes" for the printer cable that also work. The first is to cut a thin piece of electrical tape and cover pin 33 on the printer cable. This works well if the cable isn't being removed very often or for a one-time-good-deal. The other is to simply use a pair of needle nose pliers and remove pin 33. Marty's fix is the correct one, however!)

PCjr monitor tip:

Recently I picked up several IBM PCjr color monitors for $10 each. These monitors were, I understood, basically ordinary CGA monitors that also had a built in speaker and amplifier, and sported an unusual connector (18 pin dual in line pin) designed to work only with the (now LONG dead) IBM PCjr. I bought them to use them as Color Computer 3 analog RGB monitors. It turned out to be relatively easy to convert them, even though I never was able to get a schematic diagram for the monitor. Here's some of the information I managed to work out, that should help you if you ever run into one of these monitors and want to use it as either a CGA monitor or as an analog RGB monitor:

Look at the eighteen pin dual row female connector is on cable coming from IBM PCjr monitor. If you hold this cable so that it is horizontal, and you are looking into its holes, and so that the "D" stamped on one long flat side of the metal of the connector is on TOP, then the holes/connectors will be numbered as follows: Top Row, Right to Left, is numbered A1 thru A9 and the Bottom Row, Right to Left, is numbered B1 thru B9.

```
     / D \          3-D view of connector
    /     \         at the end of the cable
A9 oooooooooo A1    that comes out of the
B9 oooooooooo B1    PCjr color monitor.
```

Analog-izing the PCjr monitor:

To use the PCjr monitor as an *analog* RGB monitor, you can hook up the horizontal and vertical sync lines just as they are to the H and V sync coming out of the Color Computer 3's connector. Same for the ground lines. You can connect the audio coming out of the CoCo 3's audio RCA jack directly to the audio input on the center conductor of the audio coaxial cable that is part of the PCjr's monitor, and if you have grounded the purple wire, the hum will go away and you'll have nice use of the PCjr's speaker amplifier. The trick is to provide analog RGB inputs for the monitor. This turns out to be easier than I had feared.

Look at the circuit board of the monitor that is connected to the rear end of the picture tube. If you are looking from the rear, you'll find toward the bottom and at the left hand side of the circuit board three small transistors located right next to each other. The bases of these three small transistors connect to traces that go to one side of three 100 ohm (brown black brown) resistors located toward the bottom of the circuit board, and more or less in the middle of it. CUT the traces that feed the other side of those resistors, and hook up your R, G, and B signals from the CoCo to the top, middle, and bottom 100 ohm resistors on that other side that you've just disconnected from the CGA circuitry. That is, hook up the R, G, and B signals from the CoCo RGB port so that they go thru the 100 ohm resistors and then to the bases of those three little transistors, and so that the 100 ohm resistors no longer connect to the CGA decoder that is located on the upper right part of the circuit board.

(*editor*: This can be done with almost *any* CGA type monitor. One must pretty much know, however, where the three color driving transistors are located and which of the three are used for each color. Some circuit boards will have this information printed on them, some will not. If a schematic is available, it is a relatively easy job for an intermediate hacker.)

When you do this, the brightness and contrast controls on the front of the monitor will NOT work. However, the sub-brightness control (lower of the two adjustments on the flyback transformer in the rear of the monitor) still works fine. You can drill a hole in the back of the case in the right spot so you will have access to the sub-brightness adjustment when you've put the case back together.

PCjr Connector Pin-out

Pin #	wire color(s)	function
A1	NC	NC
A2	purple	audio (hook to ground to silence hum)
A3	NC	NC
A4	dark blue	RED CGA luminance
A5	red	GREEN CGA luminance
A6	orange	BLUE CGA luminance
A7	green	Intensity CGA
A8	NC	NC
A9	center cond. audio coax	line level audio
B1	yellow	vertical sync
B2	NC	NC
B3	brown	horizontal sync
B4	gray, light blue, & black	GROUND lines
B5	clear & pink	GROUND lines
B6	NC	NC
B7	NC	NC
B8	audio shield ground	audio shield ground
B9	main shield ground	main shield ground

NOTES:

(1) The "gray" wire and the "clear" wire look very similar, because the "gray" insulation is nearly transparant.

(2) Pins B4 and B5 are connected to several wires EACH. EG: B5 is connected to BOTH the pink coated wire and to the clear coated wire.

Just in case there are differences in how IBM assigned colored wires on different production runs of PCjr monitors, here's the assignment of colors on the connector INSIDE the PCjr monitor, where the cable connects to TWO connector sites on the board, named "C1" and "C2"

C1:		C2:	
pin #	wire color	pin #	wire color
1	red	1	brown
2	(missing / key)	2	(missing / key)
3	light blue	3	black
4	green	4	yellow
5	red	5	purple
6	dark blue	6	center cond. audio coax
7	clear	7	audio coax shield ground
8	orange		
9	gray		
10	main shield		

< 268'm >

Comments and questions may be sent in care of 68'Micros or directly to Dr. Goodman at:
1633 Bayo Vista Avenue
San Pablo, CA 94806
Delphi: martygoodman
Internet: martygoodman@delphi.com

The Glenside Color Computer Club of Illinois Presents...

The Third Annual "Last" CoCoFest!

May 21st and 22nd, 1994

at the

Holiday Inn, Elgin

A Holidome Indoor Recreation Center

345 W. River Road

RESERVATIONS: (708) 695-5000 (ask for CoCoFest rate)
RATES: $52.00 (+10% tax) per night
ADMISSION: $15.00 at door, $10.00 + SASE in advance
(or $10.00 + $0.50 postage and handling)

Send advance payments to:
George Schneeweis, Treasurer
Glenside Color Computer Club
RR #2, Box 67
Forrest, IL 61741-9629
For more info contact Tony Podraza (708-428-3576), Carl Boll (312-735-6087; CBJ@delphi.com), or Brian Schubring (708-529-3539; theschu@delphi.com)

How to get there: Elgin is about 30 miles west of Chicago. Interstqates 88,55,57, and 94 all connect to I-294. Take 294 north to I-90, then 90 west to hwy 31. Go south on 31, then take the frontage road to the Holiday Inn (east of 31). Those using I-80 & 90 east of Chicago should get on I-94 in Indiana (at I-80/90/94 intersection) and go to I-294 north. North of Chicago, take I-94 or 90 to I-294 south, then to I-90 west.

Look for the FARNA Systems booth! We will also have Ken-Ton CoCo hard drives. Hope to see you there!

Review: J&M Microtek 6809 Single Board Computer

F.G. Swygert

The J&M Microtek 6809 Single Board Computer (SBC) is a 6809 based controller card. It is designed to operate as a stand-alone computer once a ROM program is in place. Not surprisingly, it is very similar in design as the original Color Computer. This makes sense- the original CoCo was designed around "generic" Motorola specifications for a 6809 based computer.

J&M's SBC is much simpler than the CoCo, however. The entire card is only 2.75 x 5 inches. It is designed to handle 4 or 8 K of ROM and 2 or 8 K of SRAM, using 2732 or 2764 EPROMs and 6116 or 6264 SRAMs. Two 6821 Peripheral Interface Adapters (PIAs) connect the outside world to the 6809 SBC, supplying 32 bits of programmable I/O. The entire board is made up of only seven ICs (including CPU, PIAs, ROM, and SRAM) and a handful of other components. Interested parties should write J&M and ask for schematics.

There is a CPU bus consisting of a 40 pin double row male header connector. This provides all CPU and data signals for easy interfacing of peripheral boards or could be used for processor direct I/O. J&M does not currently make any special add-ons for the SBC though.

There is one (at first glance) unusual thing about this board- the schematic and parts list indicate a 4MHz or 8 MHz crystal. This crystal connects directly to the 6809's X1 and EX2 inputs. According to the 6809 data from Motorola, the on-board oscillator divides the input frequency by four to get the bus speed, so J&M has not boosted the CPU above rated speed. A standard 6809 (1MHz bus) or 68B09 (2MHz bus) can be specified- cost is the same ($60 each) for either.

The best thing I can say about this SBC is that J&M agreed to send me a bare board. This in turn was sent to Marty Goodman, who will be writing a more extensive article on it in his "Hardware Hacker" column later. Marty informs me that it should be relatively easy to make this board think it is a CoCo, so the CoCo could be used for program development. He has also indicated that increasing the memory (ROM and SRAM) should be fairly easy. We'll have to wait and see what the good doctor comes up with!

< 268'm >

The Industrial OS-9 User

Colin McKay & Doug Reid

Why OS-9000 was chosen for the CANOPUS Space Science Project.

Straight From the Horse's Mouth
by Colin McKay

First, let me introduce myself. My name is Colin McKay, and I am the Executive Vice-President of the OS-9 Users Group, Inc. I write a regular column for the Users Group Newsletter, the MOTD. The column, "Straight From the Horse's Mouth", is about the use of OS-9 in Educational, Industrial and Scientific institutions. The following is a reprint from that column, written by Doug Reid, P.Eng, an independent consultant specializing in Space Science Engineering, Flight Software, and Flight Hardware.

CANOPUS Project
Hertzberg Institute of Astrophysics
OS-9000 Synchronous Data Processor

Microware's OS-9000 Real Time Version 1.3 operating system was recently and successfully applied to the problem of processing a complex real time synchronous data stream for Canada's Hertzberg Institute of Astrophysics in support of the CANOPUS project.

CANOPUS is a space science project that monitors the magnetosphere by measuring the Earth's magnetic field, the drift velocity of the plasma in the ionosphere, and optical emissions from the aurora borealis (Northern Lights).

A number of instruments located at some 15 sites across the Canadian North send data through a 3-channel, one-way satellite communication link. The three channels are time-division multiplexed onto a single satellite link. The downlink is terminated in three synchronous modems, the output of which is a modified DDCMP (DIGITAL Data Communications Message Protocol). The link protocol defines the receive only nature of the link which disallows re-transmission of messages in which errors are detected.

Since price was an issue, OS-9000 was selected since it is targeted for off-the-shelf, relatively inexpensive, embedded PC compatible computers. Development costs were low since a PC 386 clone could be used, and the final cost of the end product was also kept in line. Another factor was that, if worse came to worse, OS-9000 is inherently upgradable to OS-9 and the subsequent high-end VME 680XX products - fortunately not necessary in this case.

For each of the six synchronous input data channels there is one HQP386CTME386/33 MHz embedded computer (from Toronto Micro Electronics Inc.), a 120 Meg IDE hard drive, an asynchronous RS-232C, and an RS-485 link back to the MS-DOS-based "Group Controller". Finally, all of the TME computers are tied together via Microware's ISP1.3 TCP/IP. The common DOS "Group Controller" is used for the user interface and display. The reliability of each channel must be "100%", so all six of the computers are on-line but each channel has a redundant channel.

The OS-9000 RunTime PAK boots from the hard drive, automatically loading and executing the application. The ongoing 9600 baud synchronous data stream is acquired via an interrupt routine that writes the data to an input buffer. Since each channel has a multiplicity of data sources, the application software acquires the data on a packet to packet basis, resyncing the data stream for each packet. Packets could be as few as one byte apart.

Once acquired, the data stream is checked for errors, filtered, reformatted, and written to an output buffer. The output buffer data is output to a VAX on a packet to packet basis via an asynchronous RS-422 at 19.2K. The protocol used to/from the VAX is a relatively simple ACK/NAK protocol. If the VAX fails, the reformatted data is written to the hard drive. Status data is dumped onto the RS-485 asynch link on command from the Group Controller approximately once per second.

The ISP1.3 is used primarily to download the archived data if and when the VAX failed. Since it was connected to the VAX Cluster, staff are able to login to a selected channel from their desks via TELNET, determine the channel's status, reconfigure the channel as necessary, and/or acquire the archived data via FTP. Over time, the DOS based Group Controller was supplanted by this more convenient interface. Changes to the application were made in the development computer and copied to the runtime processors via FTP.

The design and implementation was demonstrated through continuous on-line testing to be a success - which, in my mind, speaks volumes for OS-9000. There were, however, problems which were undoubtedly due to the fact the V1.3 felt like a relatively new product. I understand Microware's new release for OS-9000 address a lot of these problems, but I'll mention a few of them anyway, for historical reasons.

In particular, I found the built-in terminal drivers not conveniently suited for data acquisition; the description of how to define and enable an interrupt routine almost non-existent; screen control inadequate; and the debugger and RomDebug complete mysteries. I thought about creating a common driver for the synchronous channel, but gave up after considering the risks.

The V3.2 C compiler was weak and took some getting used to after my strictly ANSI C background. In short, the documentation was difficult - only the fact that Microware's Technical Support people (Pam, etc) could provide copious examples saved the day.

I should mention that Microware did strongly recommend their in-house training which undoubtedly would have alleviated many of the problems I experienced; however, in this world of fiscal restraint, those that control the purse strings would rather see me sit in my office for a month struggling than approve travel expenses. So much for common sense.

All said and done, I'm still using OS-9000 and still recommend it. Presently in our lab, a prototype 486/66Mhz running OS-9000 Professional is acquiring a contiguous synchronous 1.2Mbps data block (with a significant hardware FIFO), unscrambles the data, and writes it to a DOS hard drive for FFT analysis - not, I might add, in real time.

In addition, for all the bad mouthing I did about the comm drivers, I am presently using them to test various asynch protocols to/from an embedded processor soon to be launched on-board a NASA rocket. In short, once I got over the rather significant learning curve for OS-9000, I found it both simple and useful.

Doug Reid, PEng.

That's it for this month. If any of you have any questions or comments about the column, feel free to write me c/o the Users Group.

Colin McKay

E-mail:
cmckay@uuisis.isis.org
UUISIS - Nepean, Ontario (613) 823-6539

If you are interested in joining the OS-9 Users Group and receiving the MOTD, membership details are as follows:

United States and Canada: $25.00 US
All other countries: $30.00 US

The OS-9 Users Group, Inc.
6158 W 63rd St, Suite 109
Chicago, IL 60638
USA

Please mention the ad (page 29) and article in "68' micros" when responding to the OS-9 Users group.

< 268'm >

Operating System -Nine

NOTE: Though based on CoCo OS-9, this column is good for OS-9/68000 user as well!

Rick Ulland

Lots of odds 'n ends!.

Random Access File

If you have background tasks going, always Park your mouse. The top left corner is the fastest loop out of the hires routine, and you are spending some quality time there if looking at gshell on the console. Watch a gshell with the contrast way up- the screen flickers with the longer mouse calls.

Procedure Files Revisited

A few folks have been having similar troubles with last months procedure file file. File this file as addendum: file "procedure file file" First, remember shell's t option. Any time a shellscript acts up, insert a line with only the letter t at the very start. Shell will print each line before executing it- where the script bombs will usually be apparent. This makes a neat bootup display, by the way. Put a t line before loading system files. When done, use a -t line to stop the display. Who needs echo?.

I've given a few examples of multiple loads- separating modules with spaces on one giant load line. However, when used this way in a procedure file, there is a memory limit around 40K. The rub is, the load will complete ok! An mdir will show everything there and you'll go nuts figuring out what's wrong with the *next* line. Remember, this is an eight year old release! Try splitting a problem load into two smaller ones.

Rename is your Friend.

Ever get a non-sharable file stuck busy? Sure you have. A blown startup can't be deleted, so most editors won't touch it. They will leave a scratch file. Rename doesn't care! Call the locked file bs or delete.me or something, then slide the scratch file in place with another rename. Reboot. Fixed- once you del delete.me

How Fast is Fast?

After installing PowerBoost or Nitros9, you might want to experiment with your floppy drives interleave. Normally, a CoCo floppy drive writes a sector, then skips 3 before writing the next. This gives the cpu some time to digest, but there's no point in giving time it doesn't need. A disks interleave is set when it's formatted, and finding the best value is a matter of watching how fast the physical verify goes. With current versions of either PowerBoost or Nitros9, a floppy interleave of :2: seems to be about optimum, but this figure may change as these programs are updated.

Although slower machines can read a 'fast' disk, it will take forever- missing the first chance means waiting another 18 sectors and trying again. Keep this in mind when mailing disks, and use a number the other guy can handle.

Speaking of Boost:

As you may be finding out, patch programs have trouble with modules that already have patches in them. Burke & Burke's PowerBoost is good at working around modified systems, with one exception- the hidden files have to begin as stock.

Instead of the traditional exercise in hand fitting modules, looking for ones which modify these files, try this end around. Boot with a stock disk, swap in your modified boot, and press reset once (ctrl-alt-del on Puppo). The CoCo should go directly to the OS9Boot screen- what happens is the already loaded rel, os9p1, and stock init table are kept, just the new os9boot file is loaded. The resulting hybrid boot is then cobblered and Boosted.

These hidden files cause additional problems when making a 6809 boot on a powerboosted machine. It's simple matter to use only stock modules in the bootlist, but our friends the hidden files are going to come out of RAM- and these files are modified for the larger system stack. The new boot will fail on any machine. Same trick. Boot with stock, insert the failed boot, reset once. Now the boot should work- cobbler the result. If your new failed boot is destined to be boosted itself, simply run booster on the carcass to clean it up.

Linguistically Speaking:

The first language most folks use is Basic09- probably because it's part of the level 2 package. Which isn't to knock the language itself- it's one of the better Basics you'll find, running circles around many of the others out there. You won't hear much about Basic09 outside of the CoCo world, but you will hear the name Microware Basic- same thing. Both offer source code that can be partially processed (packed) to increase runtime speed, and either can directly access OS-9 using syscall (I have seen more than one 'basic09' program that was simply a syscall generator). The ability to directly massage OS-9 increases the power of Basic09 immeasurably.

Not much has been done to Basic09 over the years. Why mess with success? There are rumors of a 6309 version of runb in the works, otherwise it's pretty much accepted as is. The gfx2 module has been addressed by a few people, with versions up to gfx5 out there. Since these are public domain replacements, requiring users to have a copy of the alternate gfx isn't much of a problem.

Possibly the most interesting of the gfx fixes is Shawn Driscoll's guib system. Guib allows use of all sorts of window tricks like dialog boxes and radio buttons- if you intend to use a graphical user interface, this program is a real boon- makes those fancy mouse engines as easy to write as a text menu.

Most of you have seen the little merge trick usually recommended for RunB. If not, it goes like this- by merging runb, gfxs, syscall and inkey into one file, all four will be loaded and mapped together. This speeds set up time a bunch. You can do the same trick with Basic09- this hurts a little more since Basic09 is already pretty big, but writing/debugging small programs goes much faster.

As your programs get really huge, the difference in size between basic09 and source code, compared to runb and i-code, becomes important. Any modules that are completely finished can be packed, then loaded into Basic09 along with the incomplete source code modules. This buys more room, as the packed modules are much smaller.

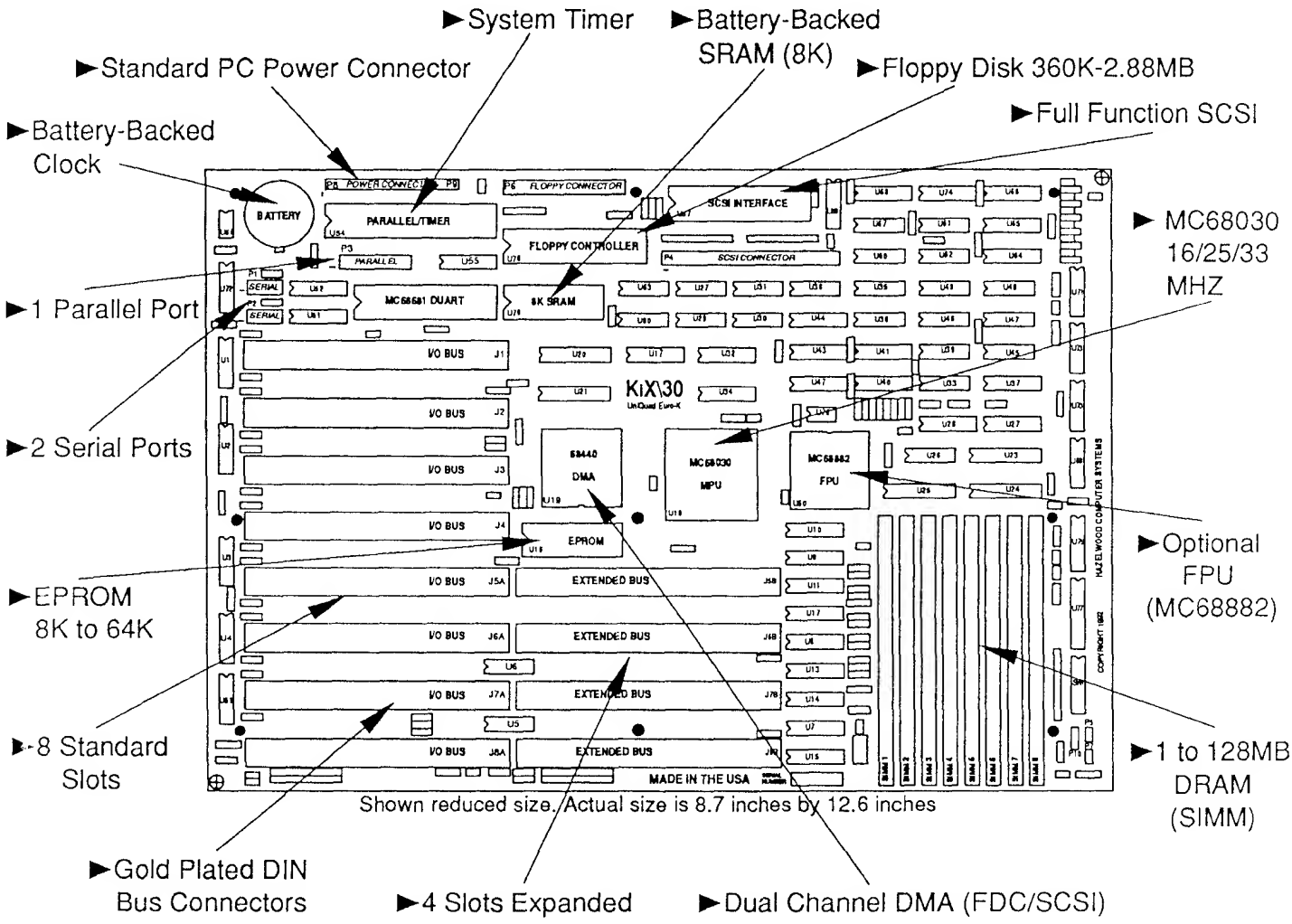
Another option is to only load the portion of the program you are working on. This is easy to visualize with menu driven programs- you may have only the menuer itself and a few modules in RAM at a time. Once all the choices have been tested individually, Basic workspace is cleaned out- no gfx2, syscall, or whatever. This gives the maximum room for loading and packing. And last, it's possible to get some more running room by reducing the data size- for example temporarily defining a 500 element array as 50 elements, then boosting the number after final testing. This isn't a terribly good idea, since code expands to fit the area available. The danger is the final version may only hold half the planned elements once the smoke clears- and ripping good code out to make data room is difficult.

As with any compiled language, it's real easy to loose the source code. Many authors have a great old program that only needs a few updates to become a great new program- only to find their only copy of the source was corrupted in the great thunderstorm of '87 or some similar fate. The executable always seems to live on as a teaser... Make lots of copies of the source- send your Mom one for Valentine's Day.

Lastly, don't let the excellence of some of our pd authors work hold you back. Your code might not look as elegant or refined as Henry Hacker's- neither does mine. Nobody really cares! So long as it preforms a useful job and doesn't release smoke from the cooling vents, it's Good Enough.

KiX/30 Highlights

Single Board Computer with Full 32-Bit Bus



Shown reduced size. Actual size is 8.7 inches by 12.6 inches

KiX\30 Motherboard

- ►4 Layer Board
- ►Mounts in PC/XT/AT Cabinet
- ►Full Documentation

Euro-K Bus

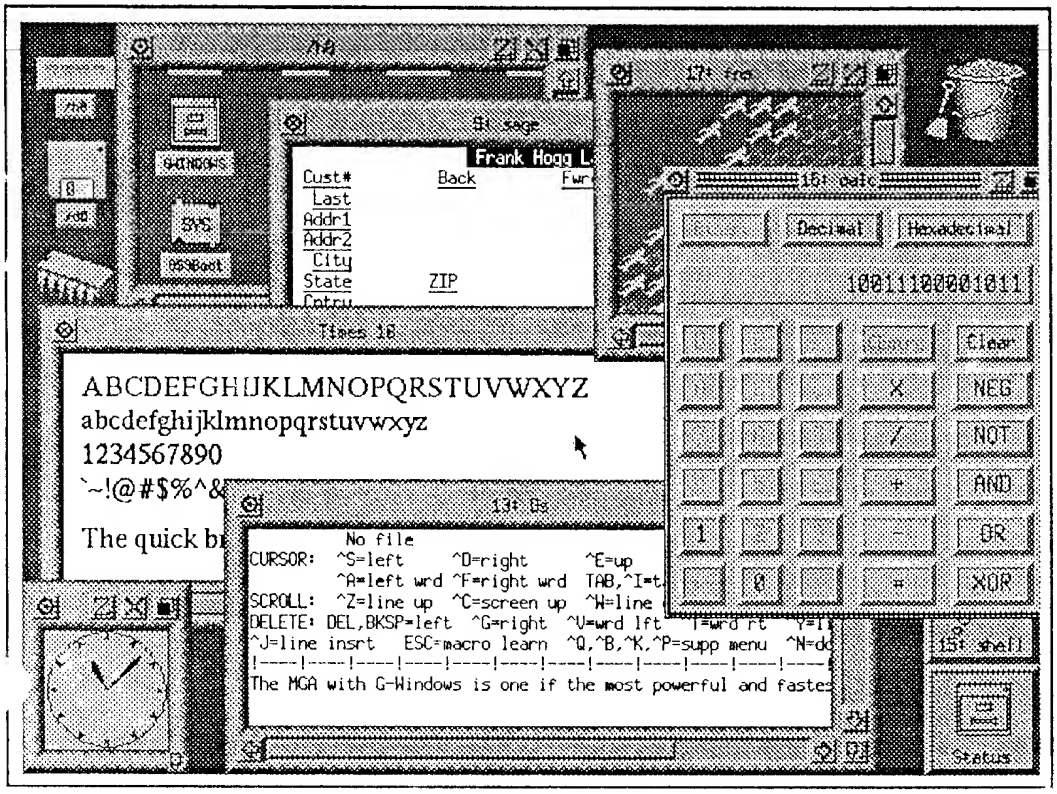
- ►Std 16 Data/Address
- ►Exp 32 Data/Address
- ►Expanded INT & DMA
- ►Board Present Logic
- ►Motorola and Intel Interfaces Support
- ►Predecoded I/O Select
- ►Cache Memory Support
- ►Open Bus Specs
- ►8/16/32 Bit Transfer
- ►Simple/Complex Modes

FRANK HOGG LABORATORY, INC.

204 Windmere Road Syracuse, NY 13205 *Fax:* 315/**469-8537** *Telephone* 315/**469-7364**

G-Windows NEW!

A powerful window-based graphical user interface for OS-9.



G-WINDOWS
provides a friendly look and feel for OS-9

The G-Desktop Manager gives a friendly look and feel to OS-9, by creating a visual representation of the OS-9 computer environment. All devices and files are represented by icons. Basic file handling functions are preformed by using point and click mouse commands.

Designed Specifically for OS-9 Users

The *G-Windows* windowing software goes far beyond anything ever offered for the OS-9 operating system. Its modular structure, multi-tasking capabilities, and unique way of seamlessly interfacing with the user's application program make it a breakthrough in the emerging technology of graphical user interfaces.

OS-9 and G-WINDOWS: A Perfect Match

G-Windows was designed specifically for OS-9 users, and takes full advantage of the best features of OS-9.

G-Windows is lean, efficient, totally ROMable, modular and fully multi-tasking.

G-Windows has an open architecture, ensuring easy growth and expansion to meet the needs of the user, today and tomorrow.

Technical Feature Summary

- Runs in high performance 68000, 20, 30 or 68040 systems.
- Truly multi-tasking. All windows may be updated at once.
- Modular Architecture. Designed as a OS-9 file manager.
- Built in VT100 terminal emulation in every screen.
- Full graphical function library built-in.
- Built in text copy and paste between windows.
- Unlimited number of available windows.
- Supports for multiple fonts, including Japanese fonts.
- Windows may be reduced to an icon to reduce screen clutter.
- Pop up menus and Alert boxes.
- Flexible color look-up table management.
- Lean and efficient. Window file manager is only 160 K.
- G-DESKTOP manager simplifies the use of OS-9.
- Totally and easily ROMable.

FRANK HOGG LABORATORY, INC.

204 Windmere Road Syracuse, NY 13205 *Fax:* 315/**469-8537** *Telephone* 315/**469-7364**

Apply Nine:

Many of the improvements to OS-9 we've discussed are provided free by their original authors- a trend somewhat unusual in computing. Perhaps it's more unusual that they are used- in the xynephobic world of IBM, such 'viri' might get the author shot.

Everybody has pd applications, its true. Usually these provide some subset of what can be done with a good (and expensive) commercial program. To some extent, this is true in OS-9 as well. But then there's telecom. It's hard to imagine a commercial telecom program for CoCo OS-9 making penny one. The pd offerings are that good. Note the entire package of separate dearchivers and download protocols forms the typical directory full of loosely related modules... it's a mess in there!

Ground Rules:

There is one minor problem with tcom in OS-9- our puny little serial port only holds one byte. That's fine for DECB but chokes OS9 a lot- there just isn't enough time between bytes to get much done. When a download is in progress, banging wildly on the window key just might bump you off-line, and don't even think of using a floppy without your terminals permission. Several folks are working on buffered serial ports to eliminate the problem, but they are going to be expensive compared to old reliable.

The patches (SACIA, 'gime toggle' clock, windint menubar patch) improve things to the point where some actual multi-tasking can be attempted during downloads. You aren't going to prove the General Theory while cleaning out Delphi's databases, but some light work in a text window isn't fatal. To add serial ports to more than one MuktiPak slot, it will have to be 'strapped', which is simply connecting all four CART (pin8) lines together.

Now That It Works:

As the "info-superhighway" starts to come of age (in other words, now that traffic jams on Delphi and Internet are commonplace), the character of telecom is changing. Gone are the hours of reading screens as they are received and replying on line. Although a few hours in conference is fun, most email traffic takes the form of immense bundled files from listservers and BBS services. You may have to do your own bundling, and waiting for 200K to come down can be an eternity when expecting news, but on-line time is costly and the sheer volume of data to wade through requires downloading.

To really push the analogy, another sign this info highway is maturing is the number of exits. Small private BBS are offering all sorts of network services, even Internet access. It's good manners to minimize the time spent on these systems so others can also use the limited number of lines available- if they can dump your mail as a file, that's more time to play!

We won't go over all the protocols available again, but there are few oddities. For instance not everyone agrees on the names. They will be:

ASCII
Xmodem
Xmodem1K or Ymodem
YModem or YBatch
ZModem

Some services support the older kermits and so forth, but this is the basic set. Note the confusion in the middle of the list. Look over the choices a BBS gives and guess.

All programs support downloading to some degree, although older programs can be kludgy. Thankfully, the older save-to-buffer system is pretty much dead, but many programs still ask for a whole lot of typing- first you tell the BBS, then you tell your terminal.

Typical of the new breed is Supercomm. Sc keeps one eye on what's being sent and received. If the file transfer menu is opened, the last word that looked like filename.ext is presented as a default (usually it's right). Should zmodem be your choice, you'll never see the menu- receiving a zmodem preamble, Sc jumps into automatic download without your assistance.

After you have your files, what to do with them? Vaughn Cato's Vu displays ASCII files, and allows scrolling through them both ways and it's very, very, very fast. Put vu in a window and a word whacker in another... happy mailtimes!

Not all is Text:

Most shareware programs arrive as archives- there may be a dozen files lurking in there. It's helpful to have a special directory tucked away to catch the mess made when these files are burst. You may see the older .arc and .pak files at times, and Carl Kreider's .ar format is everyplace- it's been popular for some time, but has begun to show it's age. The new rage in archivers for CoCo is lzh- which not only compresses better than ar but is somewhat compatible with lzh everywhere.

This is all that needs be done when dealing directly with a BBS such as Delphi. If a program has to travel further than a single BBS, there is an additional level. The ready to go archive has to be converted into a legal text file that can survive passing through unknown mailers. Commonly uuencode is used. Once received, the matching uudecode gets back to the archive, then burst into the original files.....

Then there is conference- most larger BBS support some sort of conference room where everybody's words are thrown on one big screen for the world to see. Without a 'conference mode' they are an absolute nightmare, since the screen scrolls while you are trying to type your comments.

Stupid System Trick #1:

Bang over to a different window and redirect most any command to /t2. It's output will appear in conference or be inserted in the text you are sending. How well this works depends on how heavily loaded the BBS involved is, since OS9 doesn't wait, just sends. Missed chars are missed. The display on your screen (back in the telecom program) will be completely shot in either case.

Stupid System Trick #2:

When the other guy is running to slow for SST#1 to work, fire up a graphics game in another window, in demo mode.

MVue notes-

It's possible to organize this glot of separate programs into a smooth running single screen under MVue. In a clean directory, place-

AIF.doc	text editor
AIF.com	telecom program
AIF.grb	file viewer
DEARC	directory to catch burst archives
AIF.ar	dearchivers
AIF.pak	
AIF.lzh	
AIF.arc	(etc.)

I picked the .grb extension out of the air. Since Delphi users have to do their own bundling, the user names the resulting file. Obviously if your service sends a prepackaged mail file, you have to go with the flow and match their name. There's nothing wrong with having multiple AIF for the same program, for that matter.

One 'odd' patch is required. MVue, as shipped, demands 3 letter file extensions. The only program that doesn't use 3 letters is the most popular dearciver known to CoCo (ar). Sigh.... I saw a patch once- time for some research. Remember to put -x on the parameter line of dearchiver AIFs!

The now traditional disk of the month is telcom- a health pile of archivers, terminals, and the like. As with all the pd disks, $5 delivers to anyplace in the world, though we do ask folks overseas to get a couple at once to defray the added mail cost. Since the average size of a set is over 500K, please specify the largest format you can handle.

< 268'm >

Comments and questions may be sent in care of 68'Micros or directly to Rick at:
Rick Ulland
449 South 90th
West Allis, WI 53214
(Rick is currently changing his E-mail address.)

OS-9/OSK Answers!

Joel Mathew Hegberg

Adventure programming tips in C, adjust terminal settings without tmode.

This month should be an exciting one, for this is the month many OS-9 enthusiasts anxiously await each year. What's so great about May, you ask? (Other than the rain, mud, and high pollen counts.) It's the Annual Last Chicago CoCo/OS-9 Fest, sponsored by Glenside Color Computer Club. It truely is an event to look forward to, as everyone seems to have a wonderful time. Seminars to hear, new hard-/software to check out and buy, new OS-9 computer systems, old OS-9 computer systems, and many many OS-9 enthusiasts to chat/party all night with! If you haven't reserved your room or bought your tickets yet, I encourage you to read the advertisement in this issue!

This month, I'm going to start off with a letter I received recently from Jason Bauer and end with short program written by Ted Jaeger.

Hi. You might not know me, but I met you at last year's CoCoFest in Chicago. I've been working with a C book, WorkoutC, and have started writing a program. It is an adventure game set in the ancient world. The problem I'm having is inventory. I can't find an easy way of doing inventory. The only way I can think of is a whole bunch of IF statements, but we think there has to be a better way. My dad hasn't any ideas, so he suggested I ask you.

Another question I have is: Is there any way to do "Real" random numbers. What I mean is so they don't come up in the same order each time the program is run. My book says nothing about either of these problems. Any help you could give would be greatly appreciated. Thank You.

Sincerely,
Jason Bauer.

Hi, Jason. I do remember you from the last CoCoFest, and hope we see each other again at this month's fest! (Maybe you'll let me play your adventure game then.)

Regarding the concept of 'inventory' for an adventure game (i.e. what items the player is currently carrying), there actually is an pretty elegant way of handling it. I would recommend setting up an array of bytes (type character), one byte for each item you will have in the game. So, let's say we want to have a total of five items in our entire adventure game. You could define it the following way:

```
#define NUM_ITEMS 5
char items[NUM_ITEMS];
```

When the player begins his adventure game, you probably don't want him to start out with any items in his inventory, so let's set all items in the array to zero (0x00):

```
for(i=0;i<NUM_ITEMS;i++)
items[i]=0x00;
```

What we're doing is using the items array as a boolean (ON or OFF) array, so this is a pretty simple set up. In C, any non-zero value is considered to be TRUE, while only zero is considered to be FALSE. This approach makes it easy to check whether or not the player has a needed item. For instance, lets say item #2 is 'a golden key' that the player needs in order to open a door. When the player trys to open the door, it's quite simple for your program to decide whether he can or can't open the door:

```
if(items[2])
{
  /* insert code to open door here */
}
else
  printf("Sorry, the door is locked.\n");
```

Okay, I know what you're thinking. "Yeah, this is really easy, but how about when the player requests an inventory list? There are no names associated with the game items." This really isn't hard to handle, either. I would suggest writing a subroutine to handle writing out the name of any given item number, like this:

```
print_item_name(item_number)
int item_number;
{
  switch(item_number)
   {
     case 0:
       printf("a book");
       break;
     case 1:
       printf("a rusty sword");
       break;
     case 2:
       printf("a golden key");
       break;
     case 3:
       printf("a small rock");
       break;
     case 4:
       printf("a magic ring");
       break;
   }
}
```

By making the above generic routine, you can use it whenever you need to print the name of an item, whether it be while listing the player's inventory, or while printing a room description with an item in it. Your inventory routine might look similar to this:

```
printf("Items you are carrying:\n");
for(i=0;i<NUM_ITEMS;i++)
{
  if (item[i])
   {
     print_item_name(i);
      printf("\n");
   }
}
```

Regarding random number generation, this is a very common problem with several obscure solutions. One method is to "seed" the random number generator with an arbitrary number, such as the system time. Seeding is typically done by passing a negative number to the random number generator, although this is not always the case.

What I tend to do is just call the random number generator an arbitrary number of times when my program starts up, or perhaps throughout the course of the program, and just throw away those values. This is analogous to "seeding" the random number generator. Under OS-9, you may use the current system time as the "arbitrary number" however you like. Let's say I decide to use minutes*5 +seconds as the number of times to call my random number routine at startup:

```
#include <time.h>
struct sgtbuf systime;

main()
{
  int i,number;

  getime(&systime);
 number=systime.t_minute*5+systime.t_second;
  for (i=0;i<number;i++) random();

  /* REST OF PROGRAM */
}
```

I hope that helps you out, Jason! I look forward to seeing you and your dad again, and if I can help you with something else, just let me know!

Ted Jaeger's program, 'tmd', will allow you to adjust the terminal's tmode parameters without resorting to "shell tmode etc." I'm including a ported listing for C programmers as well (listing #2). The BASIC code (listing #1) will only run on OS-9/68000 systems, due to the inavoidable use of the system-specific syscall function. The C source code, however, should work fine on any OS-9 system. Thanks for sharing your knowledge, Ted!

Listing #1: tmd for BASIC

```
PROCEDURE tmd
(* Oct 9, 1993
(* Controls screen echo and pause without
(*need of shell "tmode echo pause". Should
(* be merged with RUNB to ensure it is in
(* memory. Access from your BASIC
(* programs using something like RUN
(* tmd("-e"). With tmd your BASIC
(* programs do not need access to tmode
(* and run faster since screen control no
(* longer requires forking a shell. As an
(* extra goodie, you can disable the END
(* key to protect your programs from mad-
(* user-end-key-banging!

PARAM g:STRING[2]
TYPE registers=d(8),a(8),pc:INTEGER
DIM regs:registers
DIM c(128):BYTE

(* getstt call to look at path descriptor
(* the osk descriptor is 128 bytes
(* os9 descriptor is 32 bytes
regs.d(1)=0

regs.d(2)=0
regs.a(1)=ADDR(c)
RUN syscall($8d,regs)

(* turn off echo
IF g="-e" THEN c(5)=0
ENDIF

(* turn on echo
IF g="e" THEN c(5)=1
ENDIF

(* turn off pause
IF g="-p" THEN c(8)=0
ENDIF

(* turn on screen pausing
IF g="p" THEN c(8)=1
ENDIF

(* turn off BREAK key
IF g="-b" THEN c(18)=0
ENDIF

(* turn on BREAK key
IF g="b" THEN c(18)=5
ENDIF

(* now change the descriptor with a setstt

(* in accordance with user request
regs.d(1)=0
regs.d(2)=0
regs.a(1)=ADDR(c)
RUN syscall($8e,regs)

END
```

Listing #2: tmd for C

```
#include <sgstat.h>

/* Same features as Listing #1, but you */
/* can pass multiple options C function. */
/* tmd("e -b -p"); will turn echo on, */
/* break off, pause off. */
#include <sgstat.h>

tmd(g)
char *g;
{
  char *cptr,neg;
  struct sgbuf tmode_buffer;

  getstat(0,0,&tmode_buffer);
  neg=0x00;
  for(cptr=g;*cptr;cptr++)
   {
    if (*cptr=='-') neg=0x01;
     else
     {
       switch(*cptr)
       {
          case 'e':
                                    if (neg)
tmode_buffer.sg_echo=0x00;
                                       else
tmode_buffer.sg_echo=0x01;

             break;
          case 'p':
                                    if (neg)
tmode_buffer.sg_pause=0x00;
                                       else
tmode_buffer.sg_pause=0x01;
             break;
          case 'b':
                                    if (neg)
tmode_buffer.sg_kbach=0x00;
                                       else
tmode_buffer.sg_kbach=0x05;
             break;
       }
      neg=0x00;
     }
   }
  setstat(0,0,&tmode_buffer);
}
```

Any comments, questions, or source code to be included in Joel's column may be sent in care of 68'Micros or directly to Joel at:

Joel Mathew Hegberg
932 N. 12th Street
Dekalb, IL 60115

Delphi : JOELHEGBERG
GEnie : j.hegberg
Internet: JoelHegberg@delphi.com

< 268'm >

Darts *(continued from page 21)*

```
960 IF G=23 THEN 780
970 HPRINT (35,8),G
980 FOR X=282TO304:FOR Y=62TO72:HS
ET(X,Y,8):NEXT Y,X
990 F=(F-G)
1000 IF F=0 THEN 1160
1010 IF F>0 THEN 1070
1020 HPRINT (32,7),"BUST"
1030 FOR L=1TO100:NEXT
1040 FOR X=260TO304:FOR Y=55TO65:
HSET(X,Y,8):NEXT Y,X
1050 F=(F+G)
1060 GOTO 1090
1070 FOR X=256TO278:FOR Y=160TO17
5:HSET(X,Y,8):NEXT Y,X
1080 HPRINT(31,20),F
1090 RETURN
1100 HCLS7:HPRINT(8,5)"CONGRATUL
ATIONS No1 YOU ARE":HPRINT(15,8),"
THE WINNER"
1110 R=(R+1):PLAY"GAFAEADACABA
"
1120 HPRINT(12,12),"YOU HAVE NOW
WON":HPRINT(19,14),R:HPRINT(18,16),
"GAMES"
1130 HPRINT(5,22),"DO YOU WANT T
0 PLAY AGAIN ? (Y/N)"
1140 INPUT V$:IF V$="Y"THEN150:IF V
$="N"THEN 1210:GOTOT 1140
1150 END
1160 HCLS2:HPRINT(8,5)"CONGRATUL
ATIONS No2 YOU ARE":HPRINT(15,8),"
THE WINNER"
1170 R=(R+1):PLAY"GAFAEADACABA
"
1180 HPRINT(12,12),"YOU HAVE NOW
WON":HPRINT(19,14),U:HPRINT(18,16),
"GAMES"
1190 HPRINT(5,22),"DO YOU WANT T
0 PLAY AGAIN ? (Y/N)"
1200 INPUT P$:IF P$="Y"THEN150
1210 HSCREEN0:PALETTECMP:PALET
TE12,54:PALETTE13,0:WIDTH32:CLS
1220 PRINT" THANK YOU FOR PLAYI
NG ** D A R T S **"
1230 PRINT:PRINT:END
```

Reprinted with permission from Jan/Feb 94 issue of CoCo-Link Magazine, Australia

< 268'm >

Darts

Geoff Donges

CoCo-Link (Australia) "Encouragement Award" CoCo progamming contest winner.

Note: Several places will have notations in italics and parentheses rather than the actual characters. This seems the best way to present the listing. Don't type the italicized note, but what the note says. (i.e.- B$="*(9spaces)*" means type an asterics, 9 spaces, then an asterics.)

```
1 REM Darts for CoCo 3
2 REM (c) G. Danges, 1994
3 REM Printed w/ permission from Jan/Feb
94 CoCo-Link
10 POKE 65497,0
20 HSCREEN2:HCLS10:A$="(38 asterics)": B$=*(9 spaces)*":
HPRINT(2,0),B$:NEXT:HPRINT (2, 23),
A$:FOR X=1TO5: SOUND RND
(25 5), 1:NEXT
30 ON BRK GOTO 1210
40 HPRINT (15,10),"PROGRAMMED BY
":HPRINT(14,12),"GEOFF DONGES":HPR
INT(19,14),"1988":FOR X=1TO5:SOUND
RND(255),1:NEXT X:FOR Q=1TO500: NE
XT
50 HBUFF1,300:HBUFF4,800:HBUFF5,300
:HBUFF6,300:HBUFF7,300
60 HSCREEN0:WIDTH32:CLS
70 CLS:PRINT:PRINT:PRINT" WORLD
CHAMPIONSHIP DARTS":PRINT:PRIN
T:PRINT"TOURNAMENT":PRINT@357,
"DO YOU WANT INSTRUCTIONS"
80 INPUT O$
90 IF O$="YES"THEN 110
92 IF O$="Y"THEN 110
100 IF O$="NO"THEN 150
102 IF O$="N"THEN 150
110 CLS:PRINT"INSTRUCTIONS"
120 PRINT:PRINT"IN THIS GAME OF D
ARTS YOU PLAY A SINGLE GAME OF
[101], WITH A TALLY OF HOW MANY
GAMES YOU HAVE WON PER SESSI
ON. IT'S A STRAIGHT START/FINISH
GAME (NO DOUBLES REQUIRED). EA
CH PLAYER HAS THREE THROWS EV
EN IF HE THROWS SUCH A"
130 PRINT"BIG SCORE THAT HE [BUS
TS]. THE GAME FINISHES WHEN A PL
AYER HITS HIS CORRECT FINISH.":PR
INT:PRINT:INPUT"PRESS <ENTER> TO
PLAY";S$
140 S$=INKEY$:IF S$=""THEN 150
150
HSCREEN2:HCLS10:HCOLOR1:HCIRCL
E(80,100),20:HPAINT(80,100),7,1:HCIRCL
E(95,100,10,1,1,.25,.50:HCIRCLE(93,93),3:
HPAINT(93,93),10,1:HDRAW"BM100,100;
R2;H2":HDRAW"BM75,95LD10R5"
160 FOR C=66TO83:HCIRCLE(C,88),5:NE
XT:FOR C=90TO98:HCIRCLE(80,C),2:NE
XT:FOR C=90TO98:HCIRCLE(65,C),3:HC
IRCLE(63,C),3:NEXT C:HLINE(70,115)-(6
5,120),PSET:HLINE -(85,125),PSET:HLIN
E -(90,115),PSET
170 HLINE(65,120)-(63,125),PSET:HDRA
W"BM63,125D50R32D16L32U16BR32BD5
L32":HCIRCLE(95,160),15,1,1,.75,.25:HDRAW
"BM95,145U10R30U10L40":HPAINT(75,190),
2,1:HDRAW"BM125,133R5U2L2R2U2L2R
2U2L2R2U2L5R3":HPAINT(126,132),7,1
180 HLINE(320,0)-(320,192),PSET:HLINE
-(0,192),PSET:HCIRCLE(500,85),10,4:HLI
NE(315,53)-(315,112),PSET:HPAINT(318,6
0),4,1:HPAINT(313,90),3,1
190 HPRINT(4,0),"WORLD CHAMPIONS
HIP DARTS TOURNAMENT":HPRINT(2
0,17),"PLAYER 1 PLAYER 2":HLINE(161,
150)-(216,178),PSET,B:HLINE(241,150)-(29
3,178),PSET,B
200 HPRINT(22,20),"101"
210 C=101:F=101
220 REM
230 HPRINT(5,5),"PLAYER No 1 TO TH
ROW":HPRINT(5,7),"PRESS <ENTER> T
O THROW"
240 FOR T=1TO3:GOSUB260:NEXT T
250 GOSUB 680
260 INPUT A$:IF A$<>""THEN 270
270 HGET(135,100)-(165,115),1:HGET(75,1
40)-(85,165),2:HPUT(105,125)-(135,140),1,P
SET:HPUT(105,110)-(115,135),2,PSET
280 HDRAW"BM110,100E5H5D10E5R4U
1R5D1R5L5D1L5U1":FOR X=1TO100:NE
XT
290 FOR X=1TO100:NEXT
300 FOR X=105TO120 STEP 5:HPUT(105,
X)-(115,X+25),1,PSET:NEXT X:HGET(80,
140)-(90,160),3:HPUT(105,125)-(125,135),3,
PSET:HDRAW"BM100,125R25D10L25BR
25U1R5U2L2R2U2L2R2U2L2R2U2L5"
310 HGET(105,90)-(130,105),4
320 HGET(105,90)-(130,105),4:FOR X=105T
O283 STEP 2:HPUT(X,90)-(X+25,105),4,P
SET:NEXT
330 SOUND 25,1340 B=RND(60)
350 IF C<11 THEN B=RND(10)
360 IF B=59 THEN 340
370 IF B=58 THEN 340
380 IF B=56 THEN 340
390 IF B=55 THEN 340
400 IF B=53 THEN 340
410 IF B=52 THEN 340
420 IF B=52 THEN 340
430 IF B=49 THEN 340
440 IF B=47 THEN 340
450 IF B=46 THEN 340
460 IF B=44 THEN 340
470 IF B=43 THEN 340
480 IF B=41 THEN 340
490 IF B=37 THEN 340
500 IF B=35 THEN 340
510 IF B=31 THEN 340
520 IF B=29 THEN 340
530 IF B=23 THEN 340
540 HPRINT (35,8),B
550 FOR X=282TO304:FOR Y=62TO72:HS
ET(X,Y,8):NEXT Y,X
560 C=(C-B)
570 IF C>0 THEN 640
590 HPRINT(32,7),"BUST"PSET
600 FOR L=1TO100:NEXT
610 FOR X=260TO304:FOR Y=55TO65:HS
ET(X,Y,8):NEXT Y,X
620 C=(C+B)
630 GOTO 660
640 FOR X=177TO200:FOR Y=160TO175:
HSET(X,Y,8):NEXT Y,X
650 HPRINT (21,20),C
660 RETURN
670 GOTO 670
680 FOR M=283TO308:FOR N=90TO105:H
SET(M,N,8):NEXT N,M:FOR X=120TO130
:FOR Y=40TO50:HSET(X,Y,8):NEXT Y,X:
HPAINT(75,174),0,1:HPAINT(85,119),0,1:H
PRINT(15,5),"2":SOUND 200,5
690 FOR T=1TO3:GOSUB710:NEXT T
700 GOTO 220
710 INPUT A$:IF A$<>""THEN 720
720 HGET(135,100)-(165,115),5:HGET(75,1
40)-(85,165),6:HPUT(105,125)-(135,140),5,P
SET:HPUT(105,110)-(115,135),6,PSET
730 HDRAW"BM110,100E5H5D10E5R4U1
R5D1R5L5D1L5U1"
740 FOR X=1TO100:NEXT
750 FOR X=105TO120 STEP 5:HPUT(105,
X)-(115,X+25),5,PSET:NEXT X:HGET(80,
140)-(90,160),7:HPUT(105,125)-(125,135),7,
PSET:HDRAW"BM100,125R25D10L25BR25
U1R5U2L2R2U2L2R2U2L2R2U2L5"
760 FOR X=105TO283 STEP 2:HPUT(X,90
)-(X+25,105),4,PSET:NEXT
770 SOUND 25,1
780 G=RND(60)
790 IF F<11 THEN G=RND(10)
800 IF G=59 THEN 780
810 IF G=58 THEN 780
820 IF G=56 THEN 780
830 IF G=55 THEN 780
840 IF G=53 THEN 780
850 IF G=52 THEN 780
860 IF G=49 THEN 780
870 IF G=47 THEN 780
880 IF G=46 THEN 780
890 IF G=44 THEN 780
900 IF G=43 THEN 780
910 IF G=41 THEN 780
920 IF G=37 THEN 780
930 IF G=35 THEN 780
940 IF G=31 THEN 780
950 IF G=29 THEN 780
```

(continued on page 20)

Programming in "C"

Pointers and Arrays

P.J. Ponzo

Recall that, when we refer to &x, C will interpret this as the address in memory of the variable x. Here &x is a pointer. If we set y=&x, in our program, then y points to the variable x (y is now an address, in memory).

How do we declare a pointer variable?

In addition to the & operator (which returns the address of the variable which follows it), there is the C operator * (the asteriks).

If y points to int x, then *y is the contents of memory location y. When a program refers to *y, the C compiler will go to the address given by the pointer y, extract the integer it finds there, and use this integer in place of *y. This poses a problem... how is the compiler to know whether the contents of the address y, (which we refer to as *y, in our program) is an integer or a floating point number or a character variable???

Since an integer occupies 2 bytes (usually, depending upon the computer you are using) and a single character variable occupies just 1 byte and a float occupies 4 bytes ...etc. etc...then it's clearly important that the compiler KNOW that y is pointing to an int or a char or a float ...etc. So we must declare the type of variable that y points to! To do this we declare the contents of y, namely *y !!

```
1 main() {
2   int *y;
3   int x=123;          /* x=integer 123 */
4   y=&x;               /* y=address of x */
5   printf("The value of x is %d",*y);
6 }
```

This program will (correctly) print: The value of x is 123. Note that, in line 4, y is made a pointer, pointing to x and x is an int .... SO ... in line 2 we declare our pointer y as: int *y since it points to an int.

REMEMBER!! If " sam" is a pointer to some variable, then be sure to declare "*sam" as the same type as the variable "sam"!
If x is an int and y=&x then declare : int *y
If x is a char and y=&x then declare : char *y
If x is a float and y=&x then declare : float *y

Variables and their homes in memory

We've mentioned that the char c occupies 1 byte of memory, and that int i occupies 2 bytes and float f occupies 4 bytes.

Suppose pc=&c is a pointer to c, and pi=&i and pf=&f. Where does pc+1 point to ? Where does pi+1 point to? Where does pf+1 point to ?

```
 □  □  □  □  □  □  □  □
pc-2 pc-1 pc pc+1 pc+2 pc+3 pc+4 pc+5
 □  □  □
pc+6 pc+7 pc+8
```

If the boxes are each 1 byte of memory, and pc points to one such byte, then the scheme shown above indicates where pc-2, etc., points.

The fact that pc is a pointer to a char (because we would (should?) have declared it with char *pc) and because a char only occupies 1 byte, then C is smart enough to know that pc-2 is the address of the memory location 2 bytes below pc.

```
 □    □    □    □    □    □
pi-1  pi  pi+1  pi+2  pi+3  pi+4
```

Now pi points to a (2 byte) int, so C (whatta guy!) arranges that pi-1 points 2 bytes earlier than pi ...that way it will point to the IMMEDIATELY PRECEDING int (...of course, who knows if there IS an int stored at pc-1 ?!). Try FLOAT...

```
 □        □        □
pf      pf+1     pf+2
```

You guessed it! pf+1 will point to the VERY NEXT float, 4 bytes past where pf points (...ain't address arithmetic wonderful ?) NOTE: If we have main() {

```
float f, *pf;
f=12.3;
pf=&f;
```

...then what IS 4 bytes past the address of f ??? There is one circumstance where we WILL know "what comes after".

Hip Hip Arrays!

You may recall, from an earlier lesson, that we defined a string by declaring it to be an ARRAY of (single) characters. We'll reproduce it here:

```
char x[10];  defines an array of 10 elements.
x[0]='A';    the first element is A.
x[1]='b';    the second element is b.
x[2]='{';    the third element is {.
x[3]='\0';   the last element is 'number' 0!!!!
printf("the string is %s",x);  prints the string,
up to the 0 ...and the printout would be: the
string is Ab{ ... remember?
```

```
main() {
  char x[10], *px;
  x[0]='A';
  x[1]='b';
  x[2]='{';
  x[3]='\0';
  printf("the string is %s",x);
```

NOW, if we define px=&x[0], a pointer to the first element in the ARRAY x[], then what character does px+2 point to ?

```
{
printf("%c%c%c",*px,*(px+1),*(px+2));
```

The above would print: Ab{. printf ("%c",*px); would print the contents of memory location px, namely A, and printf("%c",*(px+1)); prints b, and printf("%c",*(px+2)); prints {, and ALL THREE GIVE: Ab{ (...a-a-a-h, C is so clever ...).

```
1  char x[4];
2  x[0]='A';
3  x[1]='b';
4  x[2]='{';
5  x[3]='\0';
```

In this program excerpt, we defined some elements of the ARRAY x[10] by saying (laboriously):x[0]='A'; x[1]='b'; x[2]='{'; x[3]='\0'; For a char ARRAY we can also say: char x[4]={ 'A','b','{','\0' }; - making Lines 2-5 unnecessary. For an int ARRAY we can say: int x[3]={11,12,13};

Under certain conditions, all arrays may be initialized (defined as they are declared). We'll talk about this later ...

More pointersand arrays

When we define an ARRAY, say: int s[10]; then each of s[0], s[1], etc are (2 byte) integers....and here's something special: s, all by itself, is a pointer to s[0]

```
1 main() {
2   int s[3];       /* define integer array */
3   s[0]=11; s[1]=12; s[2]=13;
                /* define members of s[ ] */
4   printf("%d",s[0]);
                    /* print first integer */
5   printf("\n%d",*s);
                  /* print what s points to*/
6 }
```

Line 4 will print 11 and line 5 will print 11, proving that s[0] and *s are the same, right? So s is a pointer to s[0], right? Of course it would have been easier to say: 2 int s[3]={11,12,13}; - and omit Line 3!

NOW, since an ARRAY (any ARRAY, whether ints or chars or floats) has its name as a pointer (..think about that!), then not only will s[2] give the 3rd element in the ARRAY, but so will *(s+2) give the 3rd element!

Here's a program you've seen in an earlier lesson (on STRINGS):

```
1 main() {
2   char s;             /* declare a char */
3  s="I am a string"; /* make it a string  */
4   printf("%s",s);  /* print the string  */
5  printf("\n%s",s+2);  /* what????? */
6 }
```

Line 2 declares s to be of type char.
Line 3 defines s to be a string (double quotes, remember?).
Line 4 prints the %string, giving I am a string
Line 5 prints .... am a string

Here, just as for an array, s is a pointer to the string "I am a string". In fact, s points to the first element: I. HENCE, s+2 points to the 3rd element (a) and printf() will (upon receipt of a %string address) print everything from that address up to the special terminating '\0' (...remember him?).

The name of a **string variable** is a pointer to the first element in the **string**. The name of an **array variable** is a pointer to the first element in the **array**. REMEMBE (and pointer arithmetic is wonderful)!

Be kind to your compiler

Now, if we declare char a[30]; the C-compiler knows that a refers to a collection of 30 chars ...and it knows this even before we define the elements a[0], a[1], etc.

BUT, if we declare char s;, how is the compiler to know whether s if a single char or a string of chars. If, subsequently, we say s="I am a string" then s IS a string ..but if we say s='A' then s is just one char ...but should we REALLY make the compiler figure this out? After all, the compiler would reserve a single byte in memory for just one char but may have to reserve dozens of bytes for a string of chars!

MORAL: declare a single character as "char s;", declare a string of characters as "char *s;"

Since we (and the compiler) know that the name of a string is a pointer then the declaration char *s; (followed perhaps by s="I'm a string") just anticipates the subsequent use of s as the name of a string (as opposed to a single character).

```
char s;
 s="I am a string";
 printf("%s",s);
```

Although we have used the above format in earlier lessons, *it may not compile!* And if it does, *it may not work!* If we use char *s then ALL compilers will accept it.

We have: If s is the name of a string (defined with double quotes) then s s+1 s+2 etc point to the 1st 2nd 3rd etc.; members of the string. If s is the name of an ARRAY (defined with s[0]=... etc.) then s s+1 s+2 etc point to the 1st 2nd 3rd etc.; members of the ARRAY.

Now, don't think that strings and Arrays are the same ...after all a string is only an ARRAY of (single) char variables whereas we can also have Arrays of ints or floats or other interesting data types. Further, the name of a string pointer is a VARIABLE:

```
1 main() {
2 char *x, *y, *z; /* 3 pointers to strings */
3 x="ABC";  /* x points to string ABC */
4 y="DEF";  /* y points to string DEF */
5 z=x;      /* z points to string ABC */
6  printf(" The x-string is %s",z);
7  z=y;     /* z points to string DEF */
8  printf(" The y-string is %s",z);
9 }
```

In line 5 we make z point to the same thing that x points to, so line 6 prints: The x-string is ABC. In line 7 we make z point to the same thing that y points to, and line 8 prints: The y-string is DEF. BUT, we could also say: x=y; and make x point to the y-string!

Strings and 1-dimensional character arrays

Suppose we may declare strings c and x by using:

```
1 char c[9]={'c','-','s','t','r','i','n','g','\0'};
2  char *x="x-string";
```

In the first instance, c is a pointer to an *array* of character variables (8 chars: c-string , plus the terminating '\0'). In the second instance, x is also a pointer, but now it's a pointer to a string: x-string with (again) the terminating '\0' (which C looks after appending). So what's the difference between char c[ ] and char *x ???

```
1 char c[9]={'c','-','s','t','r','i','n','g','\0'};
2  char *x="x-string";
```

In the first case, c is a *constan pointer* to the first element of the array ...and cannot be changed. In the second case, x is a *variable pointer* to the first element of the string ...and *can* be changed!

```
1 char c[9]={'c','-','s','t','r','i','n','g','\0'};
2 char *x="x-string";
3 printf(" %s  %s ",c,x);
                /* print %strings c and x */
4 x=c;          /* make x point to c */
5 printf(" %s  %s ",c,x);
                /* print %strings c and x */
```

Just to show that it is x (and NOT c) which is a VARIABLE, we declare/define them both in Lines 1 and 2. Then print them both in Line 3. We would get: c-string x-string. Then we change x, in Line 4, so it points to the same thing as c does, then print them both again, in Line 5. We'd get: c-string c-string. *You must not try: c=x; because it won't compile!* Array names point (always and forever) to the first element of the array!

More and more arrays

In addition to an array such as x[10] (with 10 elements, which may be char,int,float,etc.),here's a double array: float x[5][7]; This declares an array of 5*7=35 floating point numbers. We may define any or all of them (for example we may write x[2][1]=1.5). We think of this array as being arranged like so:

```
x[0][0]x[0][1]x[0][2]x[0][3]x[0][4]x[0][5]x[0][6]
x[1][0]x[1][1]x[1][2]x[1][3]x[1][4]x[1][5]x[1][6]
x[2][0]x[2][1]x[2][2]x[2][3]x[2][4]x[2][5]x[2][6]
x[3][0]x[3][1]x[3][2]x[3][3]x[3][4]x[3][5]x[3][6]
x[4][0]x[4][1]x[4][2]x[4][3]x[4][4]x[4][5]x[4][6]
```

Now, if we define x[2][1]=1.5, the number 1.5 goes here! And, as you might expect, we can have 3-dimensional ARRAYS, etc.

```
x[0][0]x[0][1]x[0][2]x[0][3]x[0][4]x[0][5]x[0][6]
x[1][0]x[1][1]x[1][2]x[1][3]x[1][4]x[1][5]x[1][6]
x[2][0]x[2][1]x[2][2]x[2][3]x[2][4]x[2][5]x[2][6]
x[3][0]x[3][1]x[3][2]x[3][3]x[3][4]x[3][5]x[3][6]
x[4][0]x[4][1]x[4][2]x[4][3]x[4][4]x[4][5]x[4][6]
```

When we declare a multi-dimensional array, as in float x[5][~F7], the C-compiler knows that each element occupies 4 bytes in memory (because it's an array of floats) ..hence it knows not only how much memory is needed but where each element is. Starting at the address of the first element, &x[0][0], we find the address of x[2][1] by moving 7*2+1=15 elements along the array ...and at 4-bytes-per-element that means moving 4*15=60 bytes through memory.

Notice how important the number of columns is to the C-compiler (in the declaration float x[5][7] Without this dimension the compiler wouldn't know how to get to x[2][1]! In fact, you may leave out the number of rows in declaring this array and use float x[][7] instead!

A 2-dimensional array is treated (by C) as a 1-dimensional array each of whose elements is a 1-dimensional array. The array float x[5][7] consists of 5 1-dimensional arrays, called (in C-speak), x[0], x[1], x[2], etc. and each of these 1-dimensional arrays has 7 elements.

Since (for example) the element 3 of an array sam is called sam[3] then element 3 of the (1-dimensional) array x[2] is called x[2][3]... which explains the C-notation for 2-dimensional arrays, x[i][j], rather than (the more usual) x[i,j].

In fact, you may refer to x[2][3] as (x[2])[3] ... meaning element 3 of the (1-dimensional) array x[2] (actually the 4th element since the first is element 0 !!!@#$*).

```
x[2][0]x[2][1]x[2][2]x[2][3]x[2][4]x[2][5]x[2][6]
```

If C regards x[5][7] as consisting of 5 1-dimensional arrays x[0], x[1], x[2], x[3], and x[4] then what does (for example) x[2] refer to ? What happens if we printf() x[2]? Is x[2] just a notation or does it refer to one of the 5*7=35 members of x[5][7] ?

Actually, C is very clever. x[2] (all by itself) is a pointer to the first element in the 1-dimensional ARRAY x[2], namely x[2][0]!

REMEMBER: Although x[2][0] gives the value of element x[2][0], x[2] gives the address of x[2][0]!! Since x[2] is a pointer to x[2][0], then address arithmetic can be used to obtain the address of any element in the 1-dimensional array x[2]. That means that we could identify the element x[i][j] by pointing just j elements past the address of x[i][0]. That means that x[i]+j is a pointer to x[i][j]. That means that the 'value' of this element is *(x[i]+j).

(continued on page 10)

Basic09 In Easy Steps

Chris Dekker

Differences between Disk Extended Color Basic, IBM Basic, and Basic09

As I promised in part 2, this time we will try to run some Basic09 code. However, before I get to that, I have to mention one thing: it is nearly impossible to write useful Basic09 code without the use of datastructures. Interfacing Basic09 with the OS9 operating system is also very important and nowhere near as difficult as you may think.

First though: datastructures. These are often referred to as complex datastructures and that word alone gives most people a complex (or fit). However, all of us are used to dealing with complex datastructures on a daily basis, so all you have to figure out is how to tell your CoCo what you mean. As an example I will take a line that you could findin a phone book:

Doe John, Deersville 555-1234
name address phone number

As you can see this one line (or record) consists of three parts: a name, an address and a phone number. Believe it or not this is a (simple) complex data structure!!

Now, if you want to write a program to hold addresses and phone numbers of family, friends, etc. the first thing you must do is tell the computer what a record looks like. To do so in Basic09 we use the TYPE statement: **TYPE phonerecord= name, address, phone: STRING[20]**

Once Basic09 processes this statement it knows that a record consists of three strings, each 20 characters long. The names of these three strings are (you guessed it) name, address, and phone; in that specific order. Now we must make some room for the records in the computer's memory: **DIM record(50): phonerecord.** Basic09 now knows that it must make room for 50 records and will set 50x(3x20)=3000 bytes aside for buffer space.

To Basic09 the structure we just defined looks like a blank page divided into three columns. If you want to do something useful with that page you must be able to read and write to the various records. We do this as follows.

Suppose you have defined another string variable called printname (DIM printname: STRING[20]) that you want to use to send the names on your page to the printer. To have the computer send the name "Doe John" to the printer you must use the following code:

```
printname=record(1).name
PRINT #printer,printname
```

For this code to work you must have "Doe John" in the first record (top line of the page) and you must have opened a path to your printer. Suppose you want to print all names on the page. This takes a few more lines:

```
FOR i=1 TO 50
printname=record(i).name
PRINT #printer,printname
NEXT i
```

You can access the other two strings in a record as **record(i).address** and **record(i).phone.** The line **printname=record(i).name** is basically a copy command: it copies the contents of record(i).name to printname. This copy command works two ways: it can also be used to fill the records. You do not have to worry about string overflows in Basic09. If you try to put too many characters into a string, the overflow is simply cut off. You will lose some information but your program runs fine.

Basic09 has five predefined variable types: BYTE, INTEGER, REAL, STRING and BOOLEAN. The manual does a good job explaining those (page 6-2), so I won't repeat that here. You can build up your complex datastructures from any combination of these types. Of course if you want to make your complex datastructures really complex, you can use a multi-layered design: a datastructure that consists of other complex datastructures. But that goes beyond this article, you will have to figure that one out when you get there.

Now OS-9 interfacing. This is very important because there are a lot of DECB (and MS-DOS basic) functions that are dealt with by the operating system. Consequently, you will not find an equivalent Basic09 function. But you need the information anyway so you will have to request it from OS-9 itself.

To make those requests easier Basic09 comes with a utility called SYSCALL. Syscall is a small (99 bytes) subroutine module that allows you to issue system calls from Basic09 by setting up a datastructure, pass it to syscall and run that subroutine. When it exits syscall informs Basic09 whether it was succesful or not and if so it returns the requested information.

So there you have it! Another dirty word: system calls. System calls are to OS-9 what ROM subroutines are to DECB. In fact they are ML subroutines: parts of the operating system that are accessible to the user. Just like in DECB they have entry and exit conditions. Unlike DECB you don't have to worry about USR and VARPTR functions and once you get the hang of it they are surprisingly easy to use.

The other big difference is that while the ECB manual lists 7 ROM routines (page 315) the OS-9 manual describes 132 (count em!) such routines so there must be lots of goodies in there for you.

Now let's get to work. The first program will show the use of both syscall and data structures and how they help transform Basic09 into a very powerful language. As you are probably aware, OS-9 level II (as shipped) lacks a SAVE command (or utility, if you wish) so we are going to write one right here. Just enter the following code:

```
TYPE registers=cc,a,b,dp:BYTE; x,y,u:
INTEGER
DIM regs:registers
DIM module:STRING[30]
PARAM source:STRING[30]
DIM path:BYTE; errnum, modulesize:
INTEGER
ON ERROR GOTO 100
module=source
10 regs.a=0\regs.x=ADDR(module)
RUN syscall($00,regs)
IF LAND(regs.cc,1)=1 THEN ERROR
regs.b \ ENDIF
modulesize=256*PEEK(regs.u+2)+PEEK(regs.u+3)
CREATE #path,module:WRITE+EXEC
regs.a=path\regs.x=regs.u
regs.y=modulesize
RUN syscall($8A,regs)
IF LAND(regs.cc,1)=1 THEN ERROR
regs.b \ ENDIF
CLOSE #path
RUN syscall($02,regs)
END
100 errnum=ERR
IF errnum=56 THEN
INPUT " module name? ",module
GOTO 10 \ ENDIF
PRINT " error "; errnum; " encountered"
PRINT " process aborted"
END
```

(Editor: The word INTEGER in the second and sixth lines is a continuation of the first and fifth line, INTEGER is NOT on a line by itself!)

When you are done: type q<ENTER> to exit the editor. If there are no errors (Basic09 only echoes Ready) you can run the program by simply typing run. If Basic09 reports errors you will have to go back to the editor and correct the code.

Now for the hard part: explaining what it all means. As you can see, the first line is a TYPE statement. This establishes a data structure that mimics a number of the 6809's registers. Line 2 reserves memory for the data structure. There is a difference in the use of line numbers compared with DECB: Basic09 uses them only for labels.

The next important statement is PARAM. This is used to tell the program that it can expect parameters to be passed to it on the command line. In this case if you want to write a copy of Shell to disk you would type: save("shell")<ENTER [save shell for shell+ users]. If you do not pass the program a parameter it will generate an error 56 (parameter error). This error is generated the moment you try to read the string defined by the PARAM statement. In this case: module= source. This error is dealt with in the error trapping routine starting at line number 100.

If we want to save a module it means that we are trying to send a copy of the module to a disk file. So first we must try to locate the module in memory (position independent code, remember?). To find the module OS-9 gives us a tool in the form of the LINK system call. This call is described on page 8-23 of the technical reference (page 18 of FARNA's QRG, page 23 of QRG 2).

As you can see we must set up two registers (regs.a and regs.x). In the program that happens in line 10. We set regs.a to zero, so we can save all sorts of modules, while regs.x acts as a pointer to the module name. That is the function of the ADDR statement; you will end up using ADDR a lot.

The $00 code in the next line tells OS-9 it must execute the LINK system call. This number is also given in the manual. The line after that deals with trapping errors. If a system call fails it will ALWAYS set the carry flag in the cc register and return an error code in regs.b and this is how we check for it in Basic09.

If LINK is succesful it will return the starting address of the module in regs.u (regs.u points to the first byte in the module). If you really dig deep in the manual you will find that a module's size is always contained in the 3rd and 4th byte of that module, so we can calculate it by PEEKing at offsets from regs.u. Then we must open a path to a diskfile so we can send a copy of the module to that file: CREATE#path,module:WRITE+EXEC

CREATE creates a new file.(to access an existing file use OPEN). Path is an ordinary variable. Under OS-9 it is the only thing you have to keep track of to access a file or device: there is no need to remember various numbers for devices, buffer numbers, etc. The name of the file will be the same as the name of the module because we specified module as filename. In other cases you can also use a string as file name but you must enclose that in quotation marks.

The WRITE+EXEC causes OS-9 to create a file in the execution directory (sets e attribute). If we had just usedWRITE it would have been created in the data directory. Note that we do not check for errors here. If something goes wrong Basic09 will generate the error by itself and jump to line 100.

Now we have an open path to a file and we must send the data down that path. The easiest way to do that is to use the WRITE system call. This system call is explained on page 8-64 of the technical reference (QRG page 24, QRG 2 page 28). As you can see there are three entry conditions to be met. OS-9 needs to know the path number (in regs.a), a starting point (in regs.x) and the number of bytes it must write (in regs.y). The lines right after the OPEN statement take care of that. Then we execute the WRITE call (RUN statement) and check for errors.

If all has gone well the module has been copied to disk and we can start cleaning up. First you must CLOSE the path to the disk file. The next thing we want to do is UNLINK the module we copied. For this particular program it isn't really neccessary to do so but, hey, we want to keep some standards don't we? If you check out the UNLINK system call (page 8-4, QRG 20, QRG 2 25) you will see that it's only entry condition is that regs.u points to the starting point of the module, which it still does.

That's basically all there is to it to write a SAVE utility in Basic09. Now the only thing left is the error trapping routine. You can make this as big or as small as you want. You can even leave it out altogether. In that case you will simply get an ERROR#xxx message as you will get from most OS-9 utilities.

If you want your version to be user friendly you can program a number of messages into the routine pointing out the trouble. I have taken the middle road: trap error 56 and let the others slip by. The most important reason for trapping error 56 is that I don't like typing brackets and quotation marks on a command line. This little trap solves that problem just fine.

I realize that there are a number of subjects that I have skipped (sort of), like how to actually use Basic09's editor (in part 2 we never got past the edit command). The editor functions very different from the one used by DECB, mostly on account of the absence of line numbers. Since there are only a few commands you really have to master, I won't spend much time on them here. You can find a summary on pages 4-1 and 4-2 of your manual. The commands you need most often are: +, -, s(earch), c(hange) and d(elete).

As you probably noticed the actual code is very readable as computer languages go. This is mostly due to the fact that Basic09 is very generous when it comes to naming variables. I am not sure of the exact rules, but I do know that even variable names that differ only after the 20th character are kept apart.

Even if you have sworn by now that you will never in your life write another utility, you should try to understand how data structures and the syscall utility work. The reason for that is that under OS-9 both the computer's screen and disk files are considered to be devices. So if your program needs to know something about a window (type, size, colors, etc.) or about a file (size, pointer, etc.) it MUST issue a request to OS-9 using syscall.

Now your homework (heh, heh)!! Although the above code will work just fine, it has some limitations built in. For instance- you can not save modules that were loaded as part of the bootfile. The reason for that lies in the 6809's 64K addressing space and the fact that OS-9 insists on seeing the entire bootfile as one block of code. All in all most of the time things just don't fit. To get around that we can use a SAVE utility that is written in machine code since this has much less overhead space to deal with.

I have included the code for such a utility but to be able to use it you will have to write a program that saves it to a diskfile. What the program must do is the following:

1) DIMension a buffer to hold the code (100 bytes is plenty).

2) load the code into the buffer: a loop using READ/DATA statements (which work the same as under DECB) will do fine.

3) copy the data from the buffer to a file using the WRITE system call.

Be sure to write only the exact number of bytes in the module. Also your file must be called "save" and have it's e attributes set otherwise OS-9 will not execute the program. Assuming you are working from the Basic09 disk we made in part 2 you should find the file in the CMDS directory. Now the code:

```
87 CD 00 40 00 39 11 81
5C 00 0D 01 00 34 10 86
00 10 3F 00 25 20 1F 31
30 02 EC 80 1F 02 35 10
86 02 C6 0B 10 3F 83 25
0D 1F 31 10 3F 8A 25 06
10 3F 02 25 01 5F 10 3F
06 53 61 76 E5 93 95 72
```

If you take a close look at this code you will see the sequence "10 3F" a number of times. This tells OS-9 to execute a system call. The next value is the codenumber of the call.

As you can see this program uses the same calls as the Basic09 code. The only one missing is CLOSE. We can skip that one because the EXIT call automatically closes all paths. Under Basic09 you can force an EXIT call by replacing the END statements with BYE, but not before you have debugged and saved your code. BYE also terminates Basic09 and you will lose everything you have.

< 268'm >

Review: Rusty for OS-9

F.G. Swygert

Rusty is a very neat utility collection for CoCo OS-9 users. It was written to support all the varied OS-9 disk systems, including hard drives.

The utility you are likely to use first is WRITLSN0.BAS. This is a disk basic utility that rearranges disk basic files so that track 0 is free. So why do you need to free the first track? Well, when you use the BuildDir OS-9 command to write an OS-9 directory and file descriptors on a DECB disk, making the disk into a normal OS-9 disk. BuildDir will also create a DECB directory on an OS-9 disk, making the disk readable by DECB.

The third utility in the package is RSDisk. This utility allows OS-9 access to any type of DECB disk that your drives can handle. Single files can be manipulated (copy, kill, save, directory) or all the files can be copied to an OS-9 device. RSDisk is somewhat similar to the public domain RSDOS command, but a patched CC3Disk driver in not necessary. This means you can use it with your no-halt controller.

The main utility is Rusty itself. what Rusty does is loads a CoCo BASIC or machine language program (DECB format) from an OS-9 disk device, such as a hard drive. Since many CoCo OS-9 users occasionally run DECB programs as well, this should come in very handy. And if you've stopped using DECB applications simply because you don't like swapping disks now that you have a hard drive, then Rusty is definitely for you!

What Rusty does is loads the specified DECB program into it's own process space. It then checks to see if the program is indeed a DECB program, and that nothing is loaded into DECB's memory area ($30-$3F). OS-9 is then killed and program control is transferred to DECB.

The only caveat to using Rusty is that the only way to return to OS-9 is to re-boot the computer. This also means that you can't play your favorite DECB game while doing something in another OS-9 window, since complete control now belongs to DECB.

There are some limitations as to which DECB programs will run. The following won't run under Rusty. Programs that:

-auto-load and exec.

-use multiple files.

-are more than 55.5K.

-wouldn't run on a CC3 under DECB.

What WILL run under Rusty is most stand-alone CoCo 2 and 3 programs, including word processors and telecommunications programs. Programs that are hard coded to access a certain disk drive or other device will still access that device. Programs that require multiple files (such as graphics programs) can actually be started under Rusty, but they will look for the remaining files on whichever drive they are coded to load from within the program- only the main module will run from Rusty.

Rusty's documentation is simply fantastic. Everything is fully described using as few technical terms as possible. There are sample session demonstrating the use of every one of the utilities. It is printed on 8.5"x11" paper, drilled for a standard three ring binder (not included). The manual consists of a total of 18 pages, including covers.

Only one thing is not clear in this review- how did the author (Alan DeKok) come up with the name "Rusty"? Well, the program "rusts" OS-9. Anyone who has ever tried to work on "rusty" machinery will know exactly what rust will do- bring the machinery to a halt, exactly what Rusty does to OS-9!

Rusty and accompanying utilities: $20.00 (US). Available from:

Northern Xposure
7 Greensboro Cres
Ottawa, ON, K1T 1W6
Canada
Phone 613-736-0329

The MM/1 Update

Progress being made on manufacturing the MM/1.

As this goes to press, the first MM/1 sold by BlackHawk Enterprises, Inc., will have been shipped. As financing was not forthcoming, shipping of further units will be somewhat delayed, but I have managed to get enough parts to complete one machine and this will finance the purchase of enough parts to complete my current orders. Those machines will be shipped within 2 to 3 months, provided Incor ships CPU boards on time (they generally take about 25 to 30 days to ship). I've found a source willing to assemble I/O boards for me and accept payment when I receive payment for the machine. This will be done by hand, lengthening the delivery time, but will allow financing BlackHawk Enterprises without incurring major loans. Fortunately, I currently have enough system sales to finance the manufacturing of new I/O boards and 8 Megabyte memory bus cards. I expect slow delivery rates to continue for perhaps 6 to 8 months, as I find a new manufacturer, but have every hope of having the delivery time below 30 days by the end of the year. This is possible because of our policy of maintaining an in-house inventory of Extended systems.

BlackHawk Enterprises, Inc. will be at the upcoming CoCo Fest in Chicago. We will have a supply of MIDI and Serial I/O boards as well as a couple of new software items. I'm pleased to announce KTERM 2.0 and DeskTamer 2.0. DeskTamer may be familiar to some CoCo 3 owners, and we are pleased to have been chosen by programmer Ted Jaeger to support his re-release of this fine desktop productivity tool for the CoCo 3 and the MM/1. With its notepad, phone book with dialer, appointment and event schedulers and other handy tools, you'll be able to easily tame your desktop! KTerm is a greatly enhanced version of Mike Sweets TERMINAL program. John Donaldson has done a fine job, debugging , adding color ANSI support, increasing the maximum supported baud rate to 38400, and adding support for automatic Zmodem downloads, on-line help and more. Warren Hrach tells me that the Ymodem up and download routines are among the fastest he has ever seen!

We will also have the latest operating system software for the MM/1 available for a nominal fee. Our software upgrade allows 68070 users to have access to the same improvements in serial drivers available to 68340 users, plus I've adjusted the buffers on the ports for optimal performance. SBF drivers are still not included, though this driver set is still available separately, or bundled with a SCSI tape drive for orders made before the fest. Other benefits include the latest available floppy drivers, and 1.44 meg disk descriptors, which will be the standard MM/1 diskette in the future. Windio 52 will also be included, along with sound support for both 68070 and 68340 users. Sound drivers for the 68340 are coming along, they can now PLAY sounds, and recording support is currently being added.

Also available as an upgrade are copies of Microware's Professional Documentation set. BlackHawk Enterprises, Inc. can not make these available for no cost, but they will be available at a nice discount off of the normal $195 retail price, to those who can prove purchase of an MM/1. Documentation can be shipped more quickly if paid for in advance, but COD terms are available. Look for BGFX version 4 to go on sale as soon as we receive documentation from Kevin Darling.

Another project at BlackHawk Enterprises has been sourcing for peripherals for the MM/1. By the time you read this, we will be able to offer at least one compatible monitor, as well as SCSI Hard and tape drives. We should also be able to fill your needs for printers, floppy drives, keyboards and mice. We are working on sourcing for floptical drivers, as we already have a source of drives. We intend to offer as complete support as possible for our MM/1 owners!

I'd like to think that BlackHawk has the upper hand in development for the MM/1, but we are far from alone out there, and the support from the developers in the community has been great. Projects currently underway include complete FAX modem support, upgraded word processors, new terminal programs including OSTerm/68K, and several games. I recently spoke to a new developer, who is preparing to implement a CD file manager. Bob Billson has given us his approval to distribute his forthcoming UUCP package with new systems. SubEtha Software is allowing me to include CheckBook+/OSK and a package of PD utilities with each system sold. Recent sales inquiries have included two major CD-I publishing firms. Support for the MM/1 is far from dead! In fact, the best thing going for the MM/1 is the support of a community of talented and stubborn individuals, God bless you all!! < *268'm*>

Micro News

Intel is apparently worried about the PowerPC and makers who have made inroads in the Intel compatible market. Intel is moving it's price cut schedules and introductions of new chips back. Right now the price of a 60/90MHz 3.3V Pentium is $815, nearly twice that of the PowerPC, but that price will be rolled back to $595 by the fourth quarter of this year. Systems using the processor won't be available any earlier than mid April to early May, however, as the 3.3V support chips won't be ready until then. Intel spotlighted their 33/100MHz DX4 processors on March 7th- strategically before Apple's March 14 introduction of the PowerPC Macs. Expect lower prices this summer, and "bargain" prices over the holidays. If you've been wanting an OS-9000 platform, you can probably afford to buy the OS after paying the lower prices!

The P6, a chip with three times the performance of a 60MHz Pentium, is scheduled to be ready by the beginning of 1995 rather than the originally planned 1996. That would put them about on par with the upcoming PowerPC series. Who wins the competition? The answer is obvious- the end user, regardless of which platform they choose. Prices will be lower, and performance up whichever you choose , but only one has the potential to run darn near any operating system out there...

There will be a new version of the PowerPC in 1995- the MPC615. This special version includes logic that emulates Intel x86 instructions several times faster than software emulation. Performance of the hardware emulation is expected to be equal to a 66MHz Pentium. The chip is aimed toward those who are undecided or want to "sit on the fence" between Pentium and MPC hardware. This chip is being developed independently by IBM, outside of it's partnership with Motorola and Apple. Part of the reasoning behind this development is to differentiate between IBM and non-IBM MPC systems. A bigger reason is to provide a "stop-gap" system until more MPC native software is available. The venture could backfire- if little native software is developed, machines with these chips could become (more or less) Pentium "clones". MPC native mode software developers should find the machines attractive though.

Motorola will be making a Personal Data Assistant called "Envoy" (similar to the Tandy Zoom and Apple Newton). Selling for $1,500, this is a wireless communication system designed for busy executives. There will be client software for AT&T Personal Link Services, America Online, and RadioMail wireless messaging. A two-way wireless communications module, fax modem, and PCMCIA card slot is built in.

Good news for CoCo lovers! Tom Fann informs me that GIME chips, 512K memory boards (no RAM installed), and software is still available from Tandy! Call National Parts at 1-800-442-2425 or get your local RS dealer to order. The GIME is part number MX-0992 ($34.17), the 512K board is part number 26-3336 ($39.95). There will be a $2 S&H charge and probably sales tax. Any RS CoCo software can be ordered through CMC Special Order (NOT Express Order) at greatly reduced prices! Ask your local dealer for the price of the software you want.

Apple is working toward a microkernel, multithreaded, multitasking operating system with memory protection. The new system could be called System 9, but is currently code named "Gershwin". Gershwin is similar in design concept to OS-9 (would be fitting if it were officially named System 9!) and IBM's OS/2. Apple seems to be following IBM's WorkPlace OS lead, hosting multiple operating systems on a single microkernel.

There will be three interim operating systems between Gershwin and the current System 7. The first will be System 7.12, a native PowerPC version of System 7 that will run on PowerPC Macs only. The second is code named "Mozart" and will be officially known as System 7.5. Mozart is due in mid 1994 and will run on PowerPC and 680x0 based Macs. The third, code named "Copland" and officially System 8, is due in early 1995. It will also run on PowerPC and 680x0 systems. PowerPC native versions of the Mac file system, Finder, and Apple Talk file stacks will be included- little will run under emulation mode.

According to Apple advertising, Insignia's SoftWindows, a MicroSoft Windows emulation package which uses licensed Windows code, will be available in a software bundle for all PowerPC systems. The first version of the emulator emulates a 286 and only runs in standard mode, but that will be overcome by the end of the year. The emulator is reasonably fast too, about the speed of a 486SX/25MHz or a fast 386 machine. More importantly, some long-time PC users are talking about switching to the PowerPC, people like PC Week columnist Bill Machrone (also vice president of technology for Ziff-Davis Publishing Co.). Tests show that a PowerPC Mac has two to four times the performance of 68040 and 486 based machines running the same programs when the program is written to support native mode. And the Power Macs are aggressively priced too- the "entry level" MPC601 based unit (60MHz, 8MB RAM, 160MB HD, keyboard, VGA monitor) will retail for $2,209.00. Don't expect much of a cut from volume dealers at these prices. Surprisingly, some US government agencies placed orders for PowerPC based Macs the day availability was announced. According to a top government reseller, the government is usually slower to react than private industry- but not this time. Native PowerPC (Mac) applications available now include WordPerfect 3.0 and Framemaker DTP software. Pagemaker, Freehand, Photoshop, Claris Works, and many others should be available later this year. The PowerPC Upgrade card is priced at $699.00 for the Mac IIvx and vi, Performa 600, Quadra 610 and higher, and all Centris models. An upgrade logic board ($999.00) is available for some models and may be required for some also. Contact an Apple dealer for details.

Other developments at Apple include a new highly modular motherboard to be introduced later this year. The target price for the unit is $1,500 with a 68LC040. It will be upgradeable through removal of the 68LC040 processor card to an MPC603 later. Built-in CD-ROM, TV tuner, remote control, video input, and an expansion slot will come with the unit. Customizing by changing video, processor, communications, and TV cards should be fairly easy. The new unit is targeted for the education, consumer, and small business markets. This gives Apple the opportunity to introduce the PowerPC to the consumer. All other PowerPC capable products have been business oriented and targeted.

If you want a PowerPC based laptop, go to your nearest IBM dealer and plop down a hefty $11,995.00. The seven pound machine is designed as a portable Unix workstation, not a typical laptop. Standard features include a 340MB removable HD, 16MB of RAM, 9.4 inch active matrix, thin film color LCD screen, built-in Fax-Modem, microphone, speaker, and "Track Point II" pointing device. External ports include Ethernet, SCSI II, and a PCMCIA slot for almost any other device. The internal battery life is under one hour. So now you can have your Unix and take it with you, as long as an outlet is nearby!

Two German manufacturers have announced that MPC601 based systems will be released in Europe later this year. Both will initially use the Windows NT operating system and are based on the PowerPC Reference Platform (PReP) suggested by Motorola. OS/2 for the PowerPC should be available within six months and should be available also. Windows NT hasn't been selling well on any system, partially due to it's size. Taiwan New PC Consortium is also getting into the PowerPC picture- they have an MPC601 motherboard available, also based on Motorola's PReP.

If you have new soft or hardware products, let us know! We will gladly print a free blurb for you here in MicroNews whether you advertise or not (though we will be happy to have your ad also).

FARNA Systems

Box 321
Warner Robins, GA 31099
Phone 912-328-7859
Internet: dsrtfox@ delphi.com

Software, Books, and Hardware for all OS-9/OSK Systems!

CoCo DECB Software:

CoCo Family Recorder - $17.50
Genealogy program for CoCo 3. Requires 2 drives, 80 col. monitor.

DigiTech Pro - $12.50
Sound recorder for CoCo3. Record any sound for easy play-back in your BASIC or M/L programs.

ADOS: Support for double sided drives, 40/80 tracks, faster formatting, much more!
Original (CoCo 1/2) - $15.00
ADOS 3 (CoCo 3) - $25.00
Extended ADOS 3 - $30.00
(ADOS 3 req., RAM drives, support for 512K-2MB)
ADOS 3/Ext. Combo - $50.00

Mind Games - $7.50
Collection of 9 classic games. Run from included RAM disk w/512K.

Cross Road II - $7.50
Simple Tic-Tac-Toe, but with amazing sound and graphics! Sound recorded with Digi-Tech.

Space Intruders - $14.50
Looks just like Atari's classic "Sapce Invaders"! CoCo 1/2 and 3.

Donut Dilemma - $14.50
Climb, jump, and ride elevators to top of Donut factory to shut it down! 10 levels. CoCo 1/2 and 3.

Rupert Rythym - $14.50
Collect Rupert's stolen notes, then work our correct sequence. Great action adventure!

Get Space Intruders, Donut Dilemma, and Rupert Rythym for only $33.50! Save $10!

CoCo OS-9 Software:

Patch OS-9 - $7.50
Automated program installs most popular/needed patches for OS-9 Level II. 512K and two 40T/DS (or larger) drives required. 128K 35T/SS users can get patches for manual installation.

OS-9 Point of Sale - $62.50
Maintain inventory, print invoices, customer catalog, etc. Multi-user capable under Level I or II. Supports ASCII terminals. Basic09 required. Simple menu driven interface. Includes source code!

Books:

Tandy's Little Wonder - $22.50
140 page softbound book with history and technical info for all CoCo models. Schematics, peripherals, upgrades, modifications, repairs, much more- all described in detail! Vendors, clubs, BBSs also listed.

OS-9 Quick Reference Guides
Level II (Revision 2) - $7.50
68K (based on 2.3) - $10.50
Get that bulky manual off your desk! These handy QRGs have all the most needed information in a 5.5"x 8.5" desk-top size. Includes command syntax, error codes, special keys functions, etc.

CoCo Hardware:

DigiScan Video Digitizer (formerly RASCAN) - $170
Capture images from VCR, camcorder, or TV camera. No MPI required- uses joystick ports. CoCo Max3, Max 10, Color Max 3 compatible. Special order- allow 90 days for delivery. Send $75 deposit.

FARNA Systems is now an authorized Ken-Ton dealer!
Get a plug and play, ready to run 85MB system for $600.00
Inquire for price of "no-drive" kits!

FARNA Systems Publishing Services

Type Setting and Printing: We can prepare professional looking typeset manuals, books, booklets, catalogs, and sales flyers for you - we can print or you reproduce as needed from a master set! Very reasonable prices - inquire!

Mailing Service: If you send catalogs or letter correspondence to 200 or more persons at once, we can do all work for you for about the same cost of your materials alone! How much is your time worth???

Contact Frank Swygert at above address/phone for quotes

DISTO Products

Quality hardware for your Color Computer!

2MB Upgrade (no RAM) - $99.95
Mini Disk Controller - $70
Super Controller I - $100
Super Controller II - $130
MPROM Burner - $50
3-N-1 (parallel, RS-232, RTC) - $75
SASI/SCSI Hard Disk Adapter - $75
Full Turn of the Screw - book with all Tony DiStefano's Rainbow articles, January 1983 to July 1989 - $20
Complete Schematic Set - all DISTO product schematics except 2MB upgrade - $20

Include $2 S&H for book or schematics.
Hardware S&H is $4.50 for one item, $6.50 two or more. Certified check or International Money Order only!
Running low on some items- please call for availability!
514-747-4851

DISTO

1710 DePatie,
St. Laurent, QC H4L 4A8
CANADA

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
414-258-2989 (after 5pm EST)
Internet: rickuland@ delphi.com

Mini RS-232 Port: Don't let the name fool you! This is a full featured serial port, supporting the signals needed for flow control as well as the basic 4. Jumper blocks allow readdressing or swapping DSR/DCD. No custom cables or hardware widgets needed here! Y cable users will need to add $9.95 for a power supply. **$49.95**

XPander: Don't you think the CoCo would be a lot nicer without all that mess hanging off the right side? Of course it would! Our XPander allows mounting two SCS decoded devices (like a floppy and hard drive controller) inside your CoCo. Built-in no-slot RS-232 port is similar to our "Mini" described above. The external cartridge connector is still present, and can be configured to run games or as an additional hardware slot. Kit includes new lower case shell and 12V power supply. Board only is great for use in a PC case!
Kit: $124.95 Board Only: $99.95

Hitachification: CoNect will install a Hitachi 63B09E CPU and a socket into your CoCo. Machine MUST be in working condition! The 68B09E will be returned unharmed. 90 day limited warranty. Chip and installation only **$29.95**

REPAIRS: We can repair most damaged CoCos, even those with bad traces where a 68B09 was removed. Costs vary with damage. Bad 68B09 sockets repaired for only $40! **Inquire BEFORE sending your computer.**

Quality OS-9 Software from

ColorSystems

NEW! K-Windows Chess for MM/1
Play chess on your MM/1.....$24.95

NEW! X-10 Master Control for MM/1
Use MM/1 to control you home!.....$29.95

Variations of Solitaire
Pyramid, Klondike,Spider,Poker and Canfield
MM/1.....$29.95 CoCo3.....$19.95

OS-9 Game Pack
Othello, Yahtzee, KnightsBridge, Minefield, and Battleship
MM/1.....$29.95 CoCo3.....$19.95

WPShell
An OS-9 Word Processing Point and Click Interface
CoCo3.....$14.95

Using AWK with OS-9
Includes V2.1.14 of GNU AWK for OS-9/68000
MM/1.....$14.95

To order send check or money order to:
Color Systems
P.O. Box 540
Castle Hayne, NC 28429
(916) 675-1706

Call or write for a free catalog! Demo disks also available.
NC Residents please add 6% sales tax
Owned and operated by ZacK C. Sessions

Come see us at Chicago CoCoFest!
Mention this ad and get a special discount!

EDTASM6309 Version 2.02

This is a major patch to Tandy's Disk EDTASM and offers many improvements over the original version and EDTASM6309 V.1.0. The program suppoorts all CoCo models. The CoCo 3 version uses an 80 column screen and runs in 2MHz mode. YOU MUST ALREADY OWN TANDY'S DISK EDTASM TO MAKE USE OF THIS PRODUCT. It WILL NOT work with a disk patched cartridge EDTASM.

Some Changes to EDTASM:
1) Tape no longer supported
2) Buffer increased to over 43K bytes
3) Enter V command directly from Editor & ZBUG
4) Multiple FCB & FDB data per line
5) FCS supported
6) SET command works properly
7) Screen color same as that set in Basic
8) Symbol table printed five per line (cc3 only)
9) Actual errors printed with /NL option
10) Warning on long branch where short possible
11) ZBUG defaults to numeric mode
12) Supports all RGBDOS drive numbers
13) Latest 6309 opcodes supported
14) Auto detects 6309, traps illegal opcodes

many more improvements!

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
$35 + $4 shipping & handling

Summertime is "off" season for a lot of CoCoists. If you are one of those, look forward to new releases and upgrades this fall. If you use your CoCo all year 'round, the following titles are currently available:

CoCoTop version 1.0	$24.95
CoCoTop version 1.1	$19.95
CoCoTop 1.1 + Tools 3	$34.95
OScopy/RScopy	$10.00
TOOLS 3 version 1.1	$29.95
Quickletter version 2.0	$19.95
Accounting level 2	$34.95
Investing level 2	$24.95
Level II graphics 1.2	$34.95

upgrades only $5.00 *(return original disk)*

Shipping+handling: US/Canada $3.00 all others $5. Prices in US dollars Send cheque or money order NO COD'S. Call or write for Canadian dollar prices. *Mention the name of this magazine in your order and you will receive a free bonus disk!*

C. Dekker ... *User-friendly Level II Programs!*
RR #4 Centreville, NB
E0J 1H0, CANADA
Phone 506-276-4841

The UPGRADE National Disk Magazine

Don't listen to me,,,,
From Dan Maguire (Orange Park, FL), who joined recently:
"I am writing to let you know I am very impressed with the MI&CC UPGRADE disk. .. I was blown away by your (or should I say our) club library. .. Now I am a proud member. I pledge my support."

If you want support, we're here "for you"!
Mid Iowa & Country CoCo's The UPGRADE Disk Magazine!
Now in our ninth year! Four as a national Disk Magazine.
With "The UPGRADE", we've grown to be one of the largest CoCo outreaches! Over 250 satisfied subscribers in over 40 states and 5 provinces of Canada; plus Australia & England.

Your UPGRADE subscription includes:
1. 1 year membership in MI&CC: includes the "Mid Iowa & Country CoCo" library. Select from the Best available RS & OS-9 Public Domain, Shareware, & Orphanware for only a filing/backup & mail fee. ROM burn and other support.
2. Optional Christian sub-chapter: gathers Christian oriented software for those interested.
3. UPGRADE Disk Magazine subscription: 8-10 per yr. This is a News disk magazine, not a software disk.
4. Bonus disks & double issues
5. Say you saw it in '68 Micros and receive: An UPGRADE plus a surprise bonus disk, via return mail!

UPGRADE Requires: 128K CC3, W/1 drive, RGB or TV
$16.00 US $21 Canada $31 Foreign Air
"Mid Iowa & Country CoCo" (non-profit)
Terry Simons Treas/Editor 1328 48th Des Moines, IA 50311
Include your Phone & System information

The OS-9 User's Group, Inc.

Working to support OS-9 Users

Membership includes the Users Group newsletter, MOTD, with regular columns from the President, News and Rumors, and "Straight from the Horse's Mouth", about the use of OS-9 in Industrial, Scientific and Educational institutions.

Annual Membership Dues:

United States and Canada	Other Countries
25.00 US	30.00 US

The OS-9 Users Group, Inc.
6158 W. 63d St. Suite 109
Chicago, IL 60638
USA

Northern Xposure

'Quality Products from North of the Border'

OS-9 Level II

Smash! CoCo3 Arcade Game **$29.95**
Reviewed in 01 FEB 94 "68' micros"

Thexder:OS9 **$29.95**
Send manual or rompak to prove ownership

Matt Thompson's SCSI System **$25.00**
OS-9 Level II hard disk drivers, 256 & 512 byte sector support

Rusty Launch DECB Progs from OS-9 **$20.00**

Disk Basic

Color Schematic Designer **$35.00**
Reviewed in 01AUG 93 "68' micros"

Oblique Triad Software
Various titles, order catalog for list & prices

OS-9/68000 (OSK)
Write for free catalogue

All prices in U.S. funds. Check or MO only.
Prices include S&H

7 Greenboro Cres
Ottawa, ON K1T 1W6
CANADA
(613)736-0329

MultiBoot by Terry Todd & Allen Huffman
Now have up to SIXTEEN bootfiles on your startup disk!
No more boot disk floppy swapping! MultiBoot will install itself to a cobbled boot disk and will greet you with a scrolling menu of available bootfiles!
OS-9 Req: CoCo3, OS-9 Level 2.....$19.95

Towel by Allen C. Huffman
The first EthaWin program - a disk utility for OS-9.
Use a mouse or keyboard hot-keys to perform common file and disk commands from pull-down menus. Tag multiple files for Delete, Copy, Rename, etc. and even have point 'n click disk Backup, Cobbler, Dcheck and other commands. User menu lets you specify up to seven of your own commands to execute.
OS-9 Req: CoCo3, OS-9 Level 2.....$19.95
OS/K Req: MM/1 or K-Windows Compatible.....$24.95

1992 CoCoFest SIMULATOR by Allen C. Huffman
Graphics "adventure" based on the 1992 Atlanta CoCoFest
The next best thing to having been there! Digitized graphics of the event and a text command parser (ie, "get badge", "look booth", "talk to vendor") let you see all the vendors and even run into some famous faces of the CoCo Community.
OS-9 Req: 512K CC3, OS-9 Lvl 2, 490K Disk Space...$9.95
OS/K Req: MM/1 or 100% K-Windows Compat.....$14.95

NEW: RS-DOS OS-9 Terminal Emulator by Terry Todd
Let any CoCo with a serial port act as a dedicated OS-9 terminal!
Emulates OS-9 window: screen/color codes, overlay windows, attributes, and more. Run many full-screen text applications over a null modem cable or phone line at 2400 baud through the "bit banger" port!
RS-DOS Req: CoCo3, Disk Drive.....$24.95

More items available! Contact us for a complete product listing!

Sub-Etha Software

P.O. Box 152442,
Lufkin, TX 75915
Please include $2.50 S&H per order

ADVERTISER'S INDEX: